CS 411 Operating Systems II Project 5

USB Missile Launcher

Li Li

1 What do you think the main point of this assignment is?

This work is supposed to be fun. But as the last project it also includes some . This project mainly need us to do:

- *understanding of how USB device work in the kernel* I found most of these related stuff in Chapter 13 from Third Edition of *Linux Device Drivers*, by Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman.
- *write a usb module for the usb device* In this particular case, how we implement the char device.
- *testing code* This include how we send commands to the device, then it executes. How we measure the correctness is simple: hit the target.

2 How did you personally approach the problem? Design decisions, algorithm, etc.

My attampt is: use a user-space route via libusb. And by including this libusb library, we do not have to write a module to register the USB device. However, we soon figured out that this approach is miss one important point of this assignment: write a USB module.

Then we constructed this USB moudle.For this one, we implement it as a char device. The rough structure of a this USB kernel module includes:

- *struct usb_cheeky* Structure to hold all of our device specific stuff: usb device, interface, a type for the char command.
- *usb_device_id cheeky_table* This is a table of devices that work with this driver.
- *ssize_t set_command* This function using a write syscall to write command to the USB device.
- *int cheeky_probe* It is called when a USB device is connected to the kernel. It first allocate memory for the device state and initialize it. Then saves data pointer in this interface device, creates file and use dev_info to tell the user what node is now attached to.
- *void cheeky_disconnect* This part is called when unpluggging a USB device. Bascilly, we just NULL data in the interface, remove devicefile. Kfree the dev. And print dissconnect info to dev_info.
- *struct usb_driver cheeky_driver* For this part we tell the kernel the file name and function name for the operations.
- *_int usb_cheeky_init and _exit usb_cheeky_exit* This two function is just register and deregister the device.

After finish the module, we now have to make a control program from user-space. Basiclly, here we define the command to be 8 bype. First bype is type (CONTROL_TYPE or LED_TYPE) and the specific commands: up,down,right,left,led_on,led_off,fire. In the main we pass the path of the device command file to main function as a parameter. And then we open() it. We make a command_to_ucharp() to write the command, and wirte_to_device to do the actually write. In the meanwhile, we include cursesh to get use of arrow keys. Then we turing on led light, and get commands, after fire we sleep 4 secs to finish the fire command. When hit a undifined key, we just stop. After get a q from command line we close and exit.

3 How did you ensure your solution was correct? Testing details, for instance.

When we test it, we figure out all actions should interrupt one another, except for fire which sleeps to prevent being interrupted. We mix these actions with bitwise. For instance, in case we hit KEY_RIGHT, we set the command to be like cmd &= LEFT_COMMAND, cmd |= RIGHT_COMMAND. Also, to be more precise and more easy to hit the target, the unit space every time it moves should be very small, and we only use the same bullet to fire since every bullet would likely to fire in a differnet angle.

4 What did you learn?

First of all, Linux is fun. Also as the final project, it tells me how to use a user space program to interact with given module. However, Linux is also evil, because there is too many details when you look deeper and really want to figure out what is going on there.