CSC3150 Assignment5 Report

Zhixuan LIU

118010202

1 Introduction

In this Assignment, my objective is to make a prime device in Linux, and implement file operations in kernel module to control this device. Meanwhile, I will implement ioctl function to change the device configuration. The figure below demonstrates the general idea of my solution to this project.

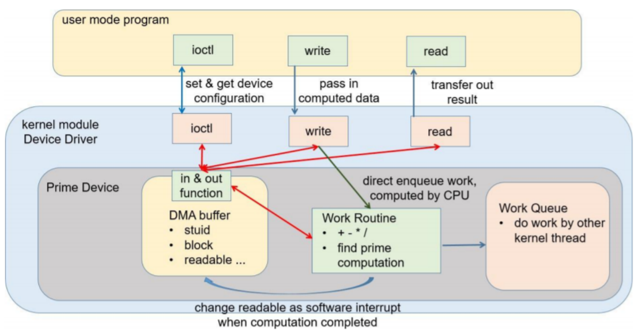


Figure 1: Genral Idea of my implementation

2 Design of my program

In this assignment, there are five functions in total that are needed to be implemented. They are drv_ioctl function, drv_read, drv_write, init_modules, and exit_modules function.

As we have user mode and kernel mode, the user's read, write and ioctl functions will conrresponds to the kernel's drv_read, drv_write, and init_modules functions.

In the following section, I will explain my design of them one by one logically.

2.1 init_module()

2.1.1 Register chrdev

In this function, some initialization jobs will be done.

First, I use dev = alloc_chrdev_region() to allocate a range of char device numbers. Then I will get an available number by MAJOR() and MINOR() macro.

After that, I allocate a cdev structure by dev_cdevp = cdev_alloc(). After which, I use cdev_init(dev_cdevp, &fops) to initialise it. "fops" is a kind of function map, which maps the function in the user mode to the function in the kernel mode.

I use cdev_add() function to make the cdev structure alive.

2.1.2 Allocate DMA buffer

Direct Memory Access buffer aka DMA buffer, is a register or memory on my device that I simulated. This buffer is as IO port mapping in the main memory. I use kzalloc() function to alloc a memory space.

2.1.3 Allocate work routine

I use kmalloc() function to allocate the work routine.

2.2 exit_module

In this function, the main purpose is to finish the process. So first, I free the DMA buffer by using kfree() function. Next, I delete character device by cdev_del() function. Furthermore, I also free the work.

2.3 drv_ioctl

In this function, it will change the IO configuration by changing the information stored in DMA buffer. Each mode corresponds to each position in DMA buffer. For example, the HW5_IOCSETSTUID changes the number store in the DMASTUIDADDR.

```
// DMA
#define DMA_BUFSIZE 64
#define DMASTUIDADDR 0x0        // Student ID
#define DMARWOKADDR 0x4         // RW function complete
#define DMAIOCOKADDR 0x8        // ioctl function complete
#define DMAIRQOKADDR 0xc        // ISR function complete
#define DMACOUNTADDR 0x10       // interrupt count function complete
#define DMAANSADDR 0x14         // Computation answer
#define DMAREADABLEADDR 0x18    // READABLE variable for synchronize
#define DMABLOCKADDR 0x1c       // Blocking or non-blocking IO
#define DMAOPCODEADDR 0x20      // data.a opcode
#define DMAOPERANDBADDR 0x21    // data.b operand1
#define DMAOPERANDCADDR 0x25    // data.c operand2
void *dma_buf;
```

Figure 2: DMA buffer structure

```
#ifndef IOC_HW5_H
#define IOC_HW5_H

#define HW5_IOC_MAGIC          'k'
#define HW5_IOCSETSTUID        _IOW(HW5_IOC_MAGIC, 1, int)
#define HW5_IOCSETRWOK         _IOW(HW5_IOC_MAGIC, 2, int)
#define HW5_IOCSETIOCOK        _IOW(HW5_IOC_MAGIC, 3, int)
#define HW5_IOCSETIRQOK        _IOW(HW5_IOC_MAGIC, 4, int)
#define HW5_IOCSETBLOCK        _IOW(HW5_IOC_MAGIC, 5, int)
#define HW5_IOCWAITREADABLE    _IOR(HW5_IOC_MAGIC, 6, int)
#define HW5_IOC_MAXNR          6

#endif
```

Figure 3: IO control modes

2.3.1 IOWAITREADABLE

This mode, the ioctl function implements kind of different things from other modes. It is used to check whether a non-blocking write has finished or not, in order to check whether the answer is readable or not.

My design is that, first I will set a readflag, which check the information in DMA buffer to see whether the mode is readable. If it is still not readable, I use a while loop to check it again. I use msleep(50) to decrease the time for while looping. If you set the msleep time much larger, the effect will be seen more clearly. For example, if you set the msleep function to 5000, the difference between block write and non-blocking write will be seen more clearly.

2.4 drv_write

This function is called when user use write function in user mode. The main contribution of this function is to put the information of the user to the DMA buffer, and let the drv_arithmetic function to figure out the answer using DMA buffer.

To begin with, I first change the readable to 0, which means the answer is not readable.

After that, the first thing to do is to put the operator and operands to the DMA buffer. Next, initialise the work. Then, check the IO mode: whether this is a blocking write or a non-blocking write. If it is a blocking write, after scheduling work, I use flush_scheduled_work function. If this is a non-blocking write, I will not use flush_scheduled_work function. Then, I set the readable mode in the DMA buffer to be 1.

After that, use drv_arithmetic function to calculate the answer of the problem.

2.5 drv_arithmetic

In this section, it do the calculation using the operator and operands stored in the DMA buffer. And also store the data to the DMA buffer. The operators include:$+-*/$. The operator is stored in the DMAOPCODEADDR, operand 1 is in the DMAOPERANDBADDR, operand 2 is in the DMAOPERANDCADDR. The result will be stored in the DMAANSADDR.

2.6 drv_read

This function is simply read the answer in the DMA buffer and also change the readable mode to be 0;

3 Execute my program

First, type "make" in the directory of main.c folder. Then, you will see many files are created.

If it is the first time to run this program, you should type "dmesg" to displays the available major and minor number, then type "sudo ./mkdev.sh major minor". This can also be removed by typing "sydo ./rmnod.sh".

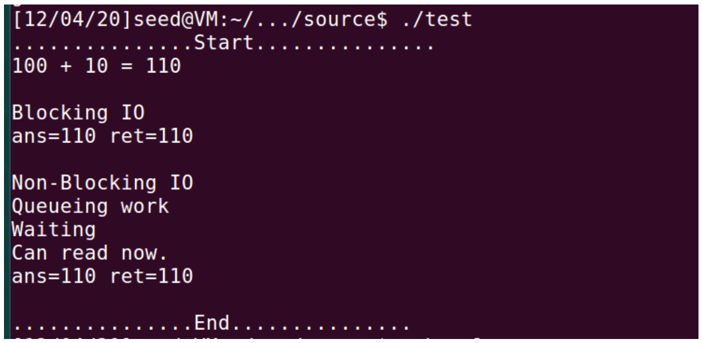
Type "./test" to run the test program.

Type "make clean". You can see the information of the kernel will be shown on the screen.

4 What I learned and problem I met

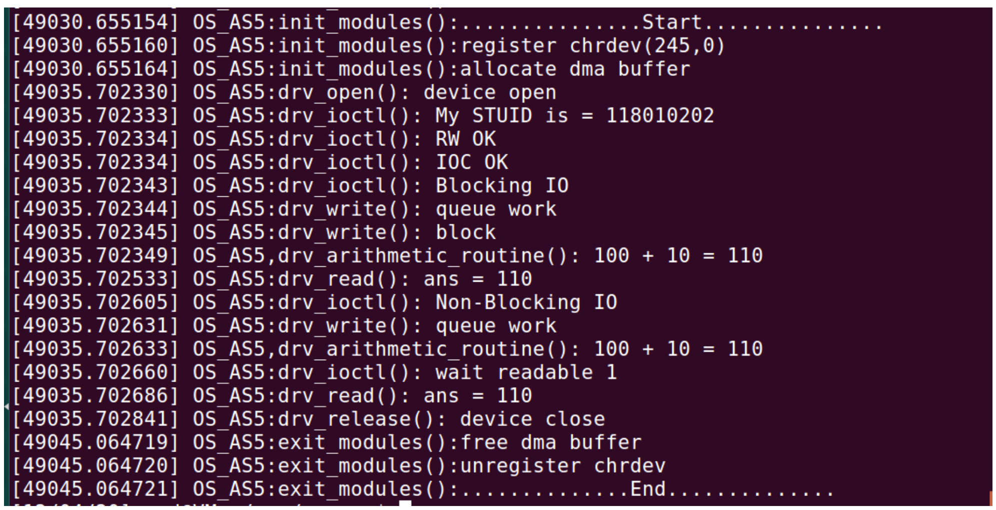
In this project, I learned how to make a prime device in Linux, and implement file operations in kernel module to control this device. I learned how to link user and kernel functions. During this project, I met a trouble about how to transfer user variable to kernel, and how to transfer kernel variables to user. Get_user function, however, does not even work when I want to transfer a structure defined by myself. I finally solve it by using copy_from_user function and copy_to_user function.

5 Screen Shots



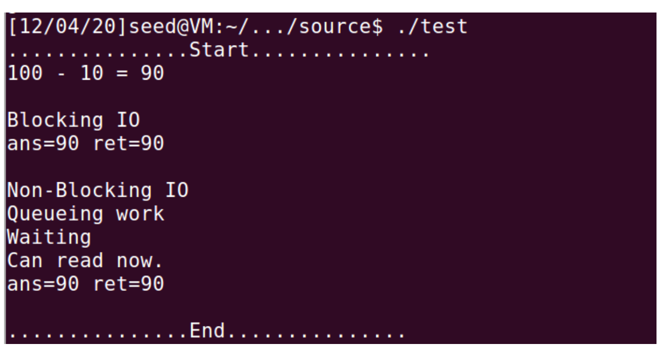
```
[12/04/20]seed@VM:~/.../source$ ./test
..............Start..............
100 + 10 = 110

Blocking IO
ans=110 ret=110

Non-Blocking IO
Queueing work
Waiting
Can read now.
ans=110 ret=110

..............End..............
```

Figure 4: operator + in user mode



```
[49030.655154] OS_AS5:init_modules():..............Start..............
[49030.655160] OS_AS5:init_modules():register chrdev(245,0)
[49030.655164] OS_AS5:init_modules():allocate dma buffer
[49035.702330] OS_AS5:drv_open(): device open
[49035.702333] OS_AS5:drv_ioctl(): My STUID is = 118010202
[49035.702334] OS_AS5:drv_ioctl(): RW OK
[49035.702334] OS_AS5:drv_ioctl(): IOC OK
[49035.702343] OS_AS5:drv_ioctl(): Blocking IO
[49035.702344] OS_AS5:drv_write(): queue work
[49035.702345] OS_AS5:drv_write(): block
[49035.702349] OS_AS5,drv_arithmetic_routine(): 100 + 10 = 110
[49035.702533] OS_AS5:drv_read(): ans = 110
[49035.702605] OS_AS5:drv_ioctl(): Non-Blocking IO
[49035.702631] OS_AS5:drv_write(): queue work
[49035.702633] OS_AS5,drv_arithmetic_routine(): 100 + 10 = 110
[49035.702660] OS_AS5:drv_ioctl(): wait readable 1
[49035.702686] OS_AS5:drv_read(): ans = 110
[49035.702841] OS_AS5:drv_release(): device close
[49045.064719] OS_AS5:exit_modules():free dma buffer
[49045.064720] OS_AS5:exit_modules():unregister chrdev
[49045.064721] OS_AS5:exit_modules():.............End.............
```

Figure 5: operator + in kernel mode



```
[12/04/20]seed@VM:~/.../source$ ./test
..............Start..............
100 - 10 = 90

Blocking IO
ans=90 ret=90

Non-Blocking IO
Queueing work
Waiting
Can read now.
ans=90 ret=90

..............End..............
```

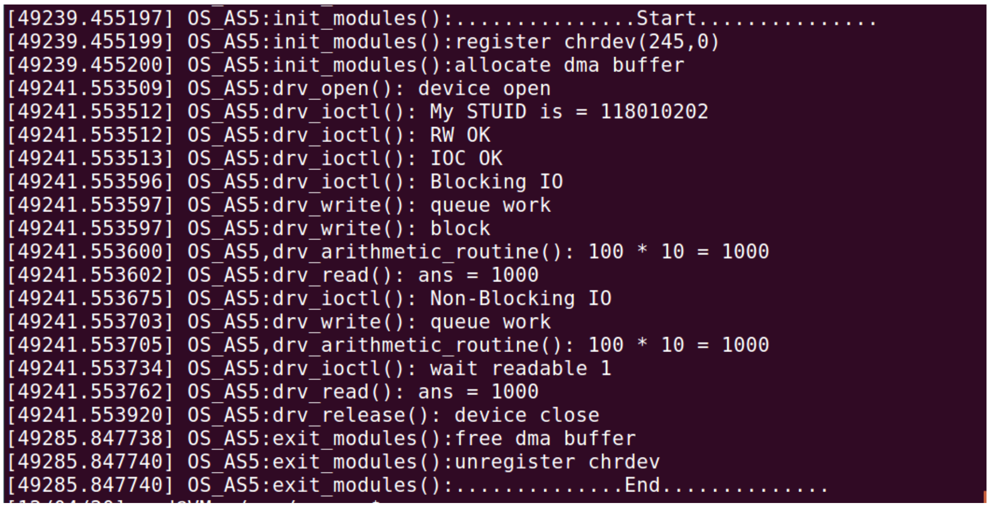
Figure 6: operator - in user mode

```
[49170.796453] OS_AS5:init_modules():...............Start..............
[49170.796455] OS_AS5:init_modules():register chrdev(245,0)
[49170.796457] OS_AS5:init_modules():allocate dma buffer
[49174.171385] OS_AS5:drv_open(): device open
[49174.171388] OS_AS5:drv_ioctl(): My STUID is = 118010202
[49174.171388] OS_AS5:drv_ioctl(): RW OK
[49174.171389] OS_AS5:drv_ioctl(): IOC OK
[49174.171465] OS_AS5:drv_ioctl(): Blocking IO
[49174.171466] OS_AS5:drv_write(): queue work
[49174.171466] OS_AS5:drv_write(): block
[49174.171469] OS_AS5,drv_arithmetic_routine(): 100 - 10 = 90
[49174.171471] OS_AS5:drv_read(): ans = 90
[49174.171535] OS_AS5:drv_ioctl(): Non-Blocking IO
[49174.171562] OS_AS5:drv_write(): queue work
[49174.171563] OS_AS5,drv_arithmetic_routine(): 100 - 10 = 90
[49174.171590] OS_AS5:drv_ioctl(): wait readable 1
[49174.171616] OS_AS5:drv_read(): ans = 90
[49174.171768] OS_AS5:drv_release(): device close
[49195.968070] OS_AS5:exit_modules():free dma buffer
[49195.968072] OS_AS5:exit_modules():unregister chrdev
[49195.968073] OS_AS5:exit_modules():.............End.............
```

Figure 7: operator - in kernel mode

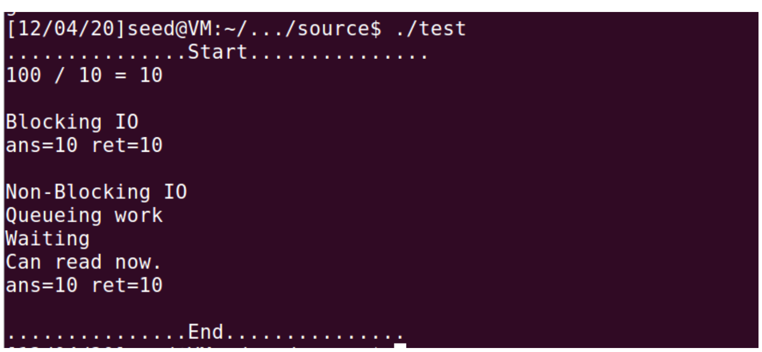
```
[12/04/20]seed@VM:~/.../source$ ./test
..............Start..............
100 * 10 = 1000

Blocking IO
ans=1000 ret=1000

Non-Blocking IO
Queueing work
Waiting
Can read now.
ans=1000 ret=1000

...............End..............
```

Figure 8: operator $*$ in user mode



```
[49239.455197] OS_AS5:init_modules():...............Start...............
[49239.455199] OS_AS5:init_modules():register chrdev(245,0)
[49239.455200] OS_AS5:init_modules():allocate dma buffer
[49241.553509] OS_AS5:drv_open(): device open
[49241.553512] OS_AS5:drv_ioctl(): My STUID is = 118010202
[49241.553512] OS_AS5:drv_ioctl(): RW OK
[49241.553513] OS_AS5:drv_ioctl(): IOC OK
[49241.553596] OS_AS5:drv_ioctl(): Blocking IO
[49241.553597] OS_AS5:drv_write(): queue work
[49241.553597] OS_AS5:drv_write(): block
[49241.553600] OS_AS5,drv_arithmetic_routine(): 100 * 10 = 1000
[49241.553602] OS_AS5:drv_read(): ans = 1000
[49241.553675] OS_AS5:drv_ioctl(): Non-Blocking IO
[49241.553703] OS_AS5:drv_write(): queue work
[49241.553705] OS_AS5,drv_arithmetic_routine(): 100 * 10 = 1000
[49241.553734] OS_AS5:drv_ioctl(): wait readable 1
[49241.553762] OS_AS5:drv_read(): ans = 1000
[49241.553920] OS_AS5:drv_release(): device close
[49285.847738] OS_AS5:exit_modules():free dma buffer
[49285.847740] OS_AS5:exit_modules():unregister chrdev
[49285.847740] OS_AS5:exit_modules():.............End.............
```

Figure 9: operator $*$ in kernel mode



```
[12/04/20]seed@VM:~/.../source$ ./test
...............Start..............
100 / 10 = 10

Blocking IO
ans=10 ret=10

Non-Blocking IO
Queueing work
Waiting
Can read now.
ans=10 ret=10

...............End..............
```

Figure 10: operator / in user mode

```
[49322.084637] OS_AS5:init_modules():...............Start...............
[49322.084639] OS_AS5:init_modules():register chrdev(245,0)
[49322.084640] OS_AS5:init_modules():allocate dma buffer
[49324.253183] OS_AS5:drv_open(): device open
[49324.253185] OS_AS5:drv_ioctl(): My STUID is = 118010202
[49324.253186] OS_AS5:drv_ioctl(): RW OK
[49324.253186] OS_AS5:drv_ioctl(): IOC OK
[49324.253193] OS_AS5:drv_ioctl(): Blocking IO
[49324.253195] OS_AS5:drv_write(): queue work
[49324.253195] OS_AS5:drv_write(): block
[49324.253200] OS_AS5,drv_arithmetic_routine(): 100 / 10 = 10
[49324.253233] OS_AS5:drv_read(): ans = 10
[49324.253312] OS_AS5:drv_ioctl(): Non-Blocking IO
[49324.253342] OS_AS5:drv_write(): queue work
[49324.253344] OS_AS5,drv_arithmetic_routine(): 100 / 10 = 10
[49324.253374] OS_AS5:drv_ioctl(): wait readable 1
[49324.253404] OS_AS5:drv_read(): ans = 10
[49324.253566] OS_AS5:drv_release(): device close
[49362.525469] OS_AS5:exit_modules():free dma buffer
[49362.525471] OS_AS5:exit_modules():unregister chrdev
[49362.525472] OS_AS5:exit_modules():.............End.............
```

Figure 11: operator / in kernel mode

```
[12/04/20]seed@VM:~/.../source$ ./test
...............Start...............
100 p 10000 = 105019

Blocking IO
ans=105019 ret=105019

Non-Blocking IO
Queueing work
Waiting
Can read now.
ans=105019 ret=105019

..............End..............
```

Figure 12: operator p in user mode

```
[49404.198961] OS_AS5:init_modules():...............Start..............
[49404.200614] OS_AS5:init_modules():register chrdev(245,0)
[49404.200616] OS_AS5:init_modules():allocate dma buffer
[49406.159117] OS_AS5:drv_open(): device open
[49406.159120] OS_AS5:drv_ioctl(): My STUID is = 118010202
[49406.159121] OS_AS5:drv_ioctl(): RW OK
[49406.159121] OS_AS5:drv_ioctl(): IOC OK
[49407.007117] OS_AS5:drv_ioctl(): Blocking IO
[49407.007119] OS_AS5:drv_write(): queue work
[49407.007120] OS_AS5:drv_write(): block
[49407.606120] OS_AS5,drv_arithmetic_routine(): 100 p 10000 = 105019
[49407.608008] OS_AS5:drv_read(): ans = 105019
[49407.608018] OS_AS5:drv_ioctl(): Non-Blocking IO
[49407.608020] OS_AS5:drv_write(): queue work
[49407.608021] OS_AS5:drv_ioctl(): wait readable 1
[49408.209243] OS_AS5,drv_arithmetic_routine(): 100 p 10000 = 105019
[49408.209881] OS_AS5:drv_read(): ans = 105019
[49408.210094] OS_AS5:drv_release(): device close
[49430.668658] OS_AS5:exit_modules():free dma buffer
[49430.668660] OS_AS5:exit_modules():unregister chrdev
[49430.668661] OS_AS5:exit_modules():..............End.............
```

Figure 13: operator p in kernel mode

```
...............Start..............
100 p 20000 = 225077

Blocking IO
ans=225077 ret=225077

Non-Blocking IO
Queueing work
Waiting
Can read now.
ans=225077 ret=225077

...............End..............
```

Figure 14: operator p in user mode

```
[49454.440688] OS_AS5:init_modules():...............Start...............
[49454.440690] OS_AS5:init_modules():register chrdev(245,0)
[49454.440691] OS_AS5:init_modules():allocate dma buffer
[49456.947923] OS_AS5:drv_open(): device open
[49456.947925] OS_AS5:drv_ioctl(): My STUID is = 118010202
[49456.947926] OS_AS5:drv_ioctl(): RW OK
[49456.947926] OS_AS5:drv_ioctl(): IOC OK
[49460.442806] OS_AS5:drv_ioctl(): Blocking IO
[49460.442808] OS_AS5:drv_write(): queue work
[49460.442809] OS_AS5:drv_write(): block
[49463.032604] OS_AS5,drv_arithmetic_routine(): 100 p 20000 = 225077
[49463.036143] OS_AS5:drv_read(): ans = 225077
[49463.036155] OS_AS5:drv_ioctl(): Non-Blocking IO
[49463.036156] OS_AS5:drv_write(): queue work
[49463.036158] OS_AS5:drv_ioctl(): wait readable 1
[49465.637152] OS_AS5,drv_arithmetic_routine(): 100 p 20000 = 225077
[49465.637755] OS_AS5:drv_read(): ans = 225077
[49465.637927] OS_AS5:drv_release(): device close
[49487.921006] OS_AS5:exit_modules():free dma buffer
[49487.921008] OS_AS5:exit_modules():unregister chrdev
[49487.921008] OS_AS5:exit_modules():..............End..............
```

Figure 15: operator p in kernel mode