Report for CSC3150 assignment 5

舒欣 118020362

How did I design my program?

Overall, I think this assignment does not have much code that we need to write by ourselves, the most difficult part is to understand what is this program doing.

About drv_read():

The thing that this function do is just to read the computed result from DMA buffer to user space. Since read is not like write, it only has blocking mode (since for non-blocking write case, if the user wants to read something, he or she needs to first check whether it is readable). Therefore, it will wait until the work function (which is drv_arithmetic_routine()) complete computation and result update (which is the variable *ans* in my program), then the read function (which is drv_read) will be called and it can guarantee that the result read is correct. After that we need to clear the result for next usage. So I use myini() to get the ans from DMAANSADDR, and use put_user() to transfer it from kernel space to user space, finally clear the result and readable variable, indicates that the result is none and it is not readable.

About drv_write():

The things that this function do is to read in the DataIn struct *data* into DMA buffer and initialize and schedule work routine based on blocking or non-blocking writing. So I first allocate a space in kernel space to store the DataIn struct read from user space using copy_from_user(). Then I use myin functions to write opcode, operand1, operand2 into DMA buffer. Then I initialize the work routine according to blocking and non-blocking write mode.

About drv_ioctl():

The things that this function do is to read and write DMA buffer. From ioc_hw5.h, we can know that HW5_IOCSETSTUID, HW5_IOCSETRWOK, HW5_IOCSETIOCOK, HW5_IOCSETIRQOK, HW5_SETBLOCKOK are where drv_ioctl() write data in (since they are all _IOW), HW5_IOCWAITREADABLE is where drv_ioctl() read data from (since it is _IOR). So I use switch case to read/write data in corresponding address according to the command. If the input command is not one of the cases I listed, just return -1, indicating that this is not a valid command. About implementing the synchronize function for non-blocking write, I just use a while loop to wait for the variable *readable* to be 1 (which means could read) in HW5_IOCWAITREADABLE case, since there is another kernel thread executing the work function (drv_arithmetic_routine()) and when the execution is completed, the *readable* variable will change from 0 (which means not readable) to 1. Only at that time we can read in the correct result.

About drv_arithmetic_routine():

This function is the work function. What it does is to compute and store the result in DMA buffer. The computation part's code could just be copied from template codes you give us. What's important in this function is the control of the variable *readable.* And this is also very simple, just set readable to 0 when it starts to compute and set it to 1 after we finish computing and writing result.

About init_modules():

This function is to initialize kernel module and char device and bind them together. I first use alloc_chrdev_region() to allocate a range of char device numbers, then get available number by MAJOR() and MINOR() macro. Then I allocate a cdev structure by cdev_alloc() and initialize it by cdev_init() in order to bind file operations with it. Finally, I add it to kernel module and make it alive by cdev_add(). Moreover, I allocate a DMA buffer and a work_routine struct in this function.

About exit_modules():

I first unregister and delete the char device I created in init_module(), then I free memory for DMA buffer and work routine.

What problems I met in this assignment and how did I solve it?

1. Originally, I do not understand what does the synchronize function for nonblocking write mean, then I see in test.c that for non-blocking write, after write (read in data and compute) function, it only has ioctl(fd, HW5_IOCWAITREADABLE, &readable), then directly check readable and if it is 1, the user can read the result, so I guess the while loop that wait for *readable* to be 1 must lay in HW5_IOCWAITREADABLE case of ioctl() function, and finally succeed!

2. When transferring DataIn struct from user space to kernel space, get_user() cannot work. So I allocate a space in kernel to store the read-in DataIn struct, and make a pointer point to it, then pass it to copy_from_user(), it works!

The steps to run my program:

First, you load my source folder in your virtual machine under random directory (if it does not run successfully, please move my source folder under the work folder of your vm). Then in terminal, you go to my source folder directory, type the following commands:

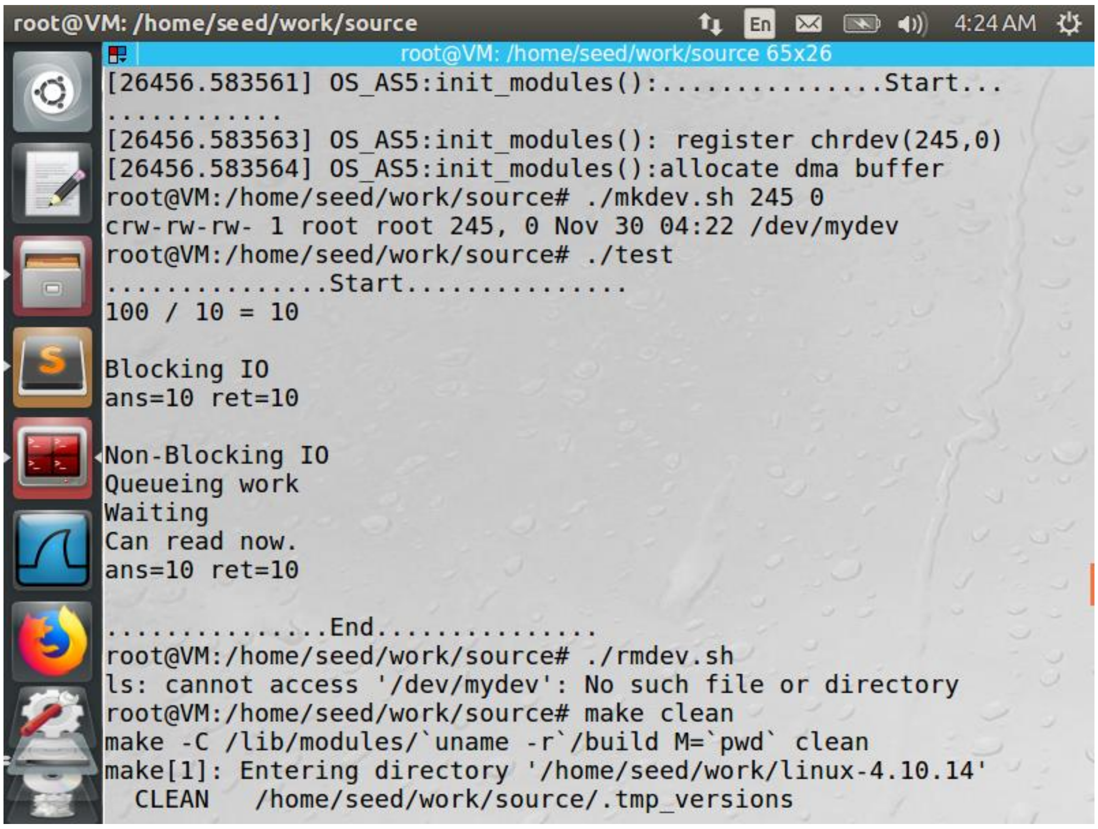
1. sudo su
2. make
3. dmesg | tail -3 (get MAJOR_NUM and MINOR_NUM)
4. chmod u+r+x mkdev.sh
5. ./mkdev.sh MAJOR_NUM MINOR_NUM
6. ./test
7. chmod u+r+x rmdev.sh
8. make clean

you could see the output and the kernel log at 6th, 8th steps, respectively.

Screenshots of my program:

Test case 1:

Output:



root@VM: /home/seed/work/source
En
4:24 AM
root@VM: /home/seed/work/source 65x26

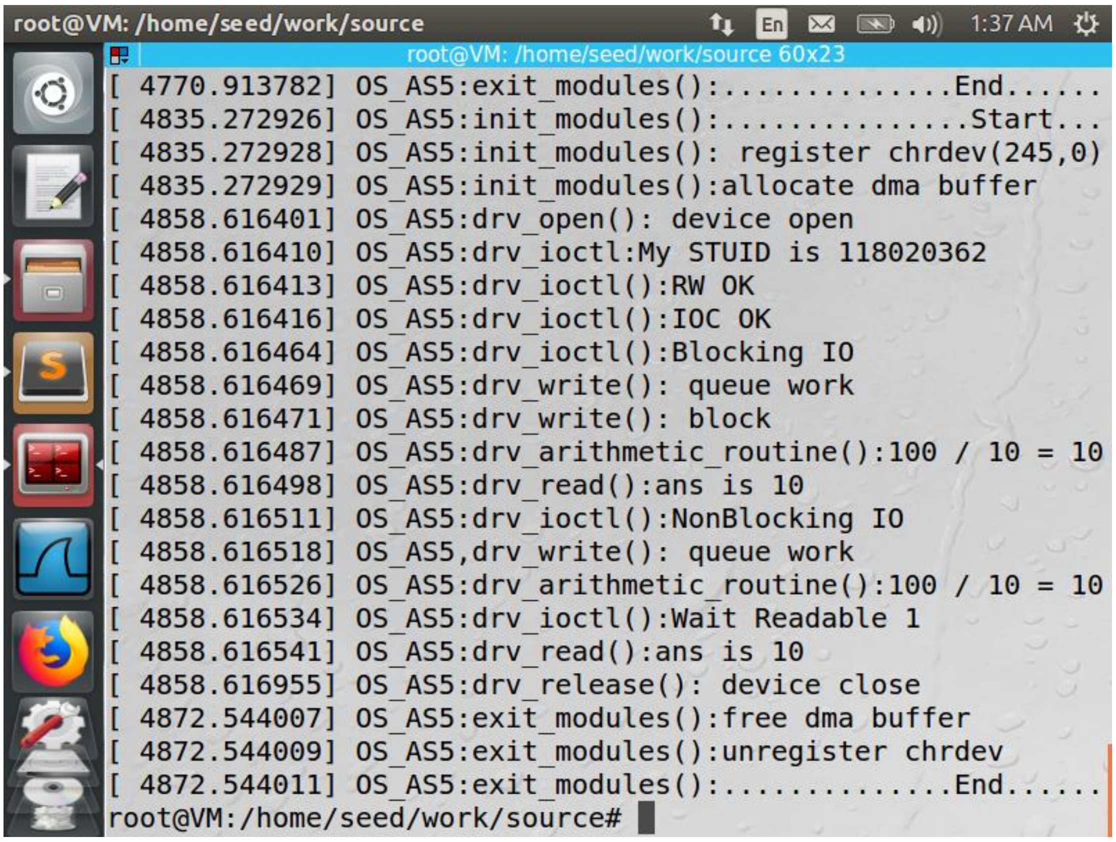
```
[26456.583561] OS_AS5:init_modules():..............Start...
...........
[26456.583563] OS_AS5:init_modules(): register chrdev(245,0)
[26456.583564] OS_AS5:init_modules():allocate dma buffer
root@VM:/home/seed/work/source# ./mkdev.sh 245 0
crw-rw-rw- 1 root root 245, 0 Nov 30 04:22 /dev/mydev
root@VM:/home/seed/work/source# ./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..............
root@VM:/home/seed/work/source# ./rmdev.sh
ls: cannot access '/dev/mydev': No such file or directory
root@VM:/home/seed/work/source# make clean
make -C /lib/modules/`uname -r`/build M=`pwd` clean
make[1]: Entering directory '/home/seed/work/linux-4.10.14'
  CLEAN   /home/seed/work/source/.tmp_versions
```

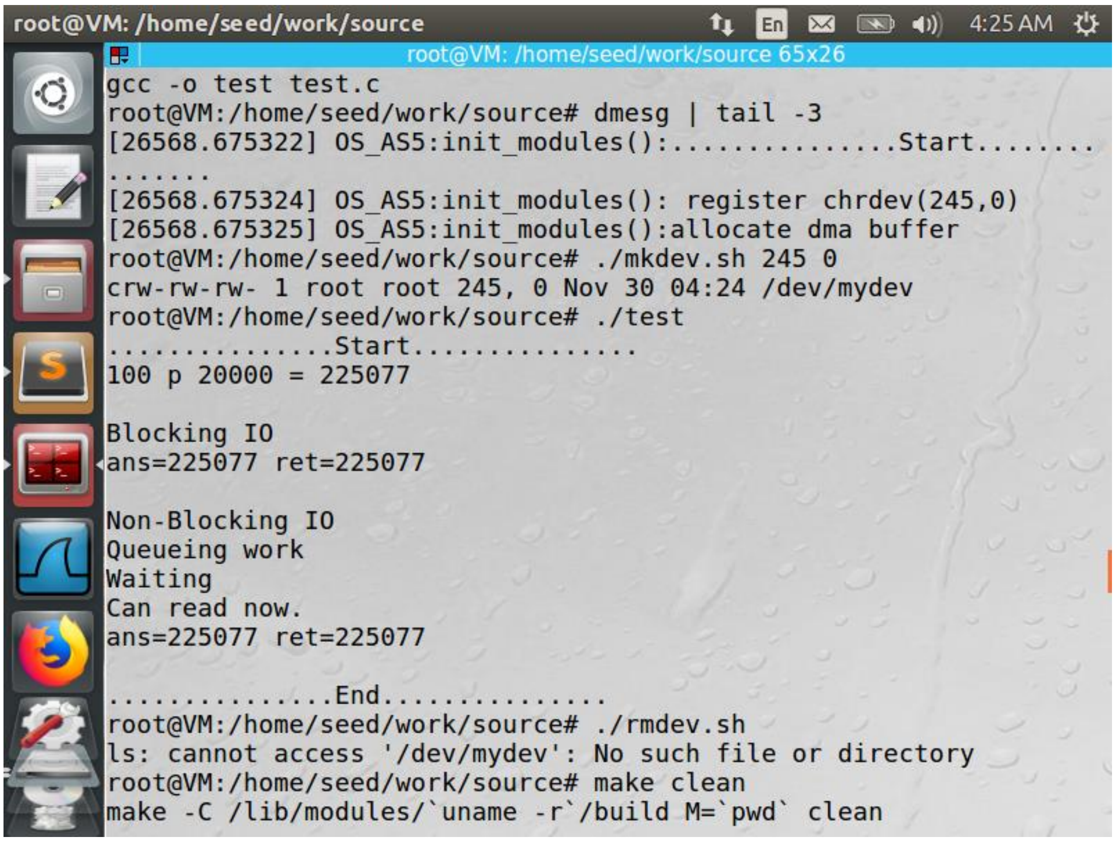
Kernel log:



root@VM: /home/seed/work/source
En
1:37 AM
root@VM: /home/seed/work/source 60x23

```
[ 4770.913782] OS_AS5:exit_modules():.............End......
[ 4835.272926] OS_AS5:init_modules():..............Start...
[ 4835.272928] OS_AS5:init_modules(): register chrdev(245,0)
[ 4835.272929] OS_AS5:init_modules():allocate dma buffer
[ 4858.616401] OS_AS5:drv_open(): device open
[ 4858.616410] OS_AS5:drv_ioctl:My STUID is 118020362
[ 4858.616413] OS_AS5:drv_ioctl():RW OK
[ 4858.616416] OS_AS5:drv_ioctl():IOC OK
[ 4858.616464] OS_AS5:drv_ioctl():Blocking IO
[ 4858.616469] OS_AS5:drv_write(): queue work
[ 4858.616471] OS_AS5:drv_write(): block
[ 4858.616487] OS_AS5:drv_arithmetic_routine():100 / 10 = 10
[ 4858.616498] OS_AS5:drv_read():ans is 10
[ 4858.616511] OS_AS5:drv_ioctl():NonBlocking IO
[ 4858.616518] OS_AS5,drv_write(): queue work
[ 4858.616526] OS_AS5:drv_arithmetic_routine():100 / 10 = 10
[ 4858.616534] OS_AS5:drv_ioctl():Wait Readable 1
[ 4858.616541] OS_AS5:drv_read():ans is 10
[ 4858.616955] OS_AS5:drv_release(): device close
[ 4872.544007] OS_AS5:exit_modules():free dma buffer
[ 4872.544009] OS_AS5:exit_modules():unregister chrdev
[ 4872.544011] OS_AS5:exit_modules():.............End......
root@VM:/home/seed/work/source#
```

Test case 2:



root@VM: /home/seed/work/source
En
4:25 AM
root@VM: /home/seed/work/source 65x26

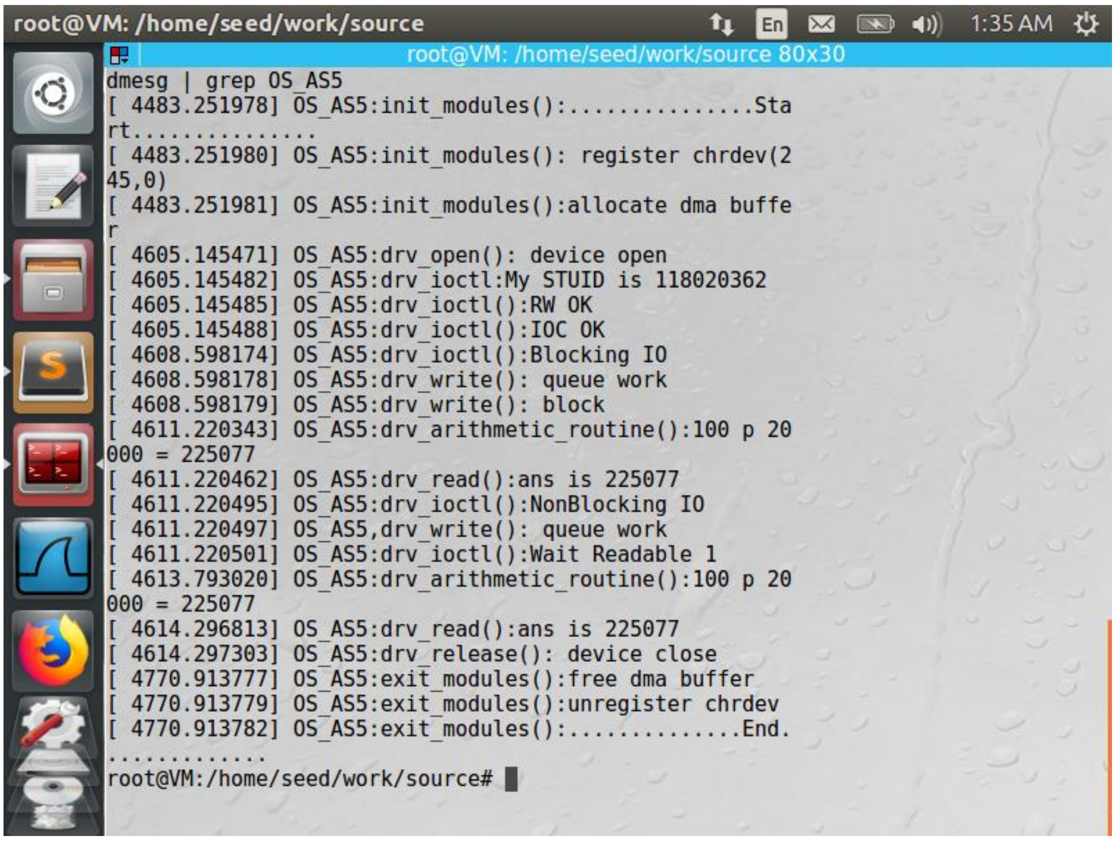
```
gcc -o test test.c
root@VM:/home/seed/work/source# dmesg | tail -3
[26568.675322] OS_AS5:init_modules():..............Start........
.......
[26568.675324] OS_AS5:init_modules(): register chrdev(245,0)
[26568.675325] OS_AS5:init_modules():allocate dma buffer
root@VM:/home/seed/work/source# ./mkdev.sh 245 0
crw-rw-rw- 1 root root 245, 0 Nov 30 04:24 /dev/mydev
root@VM:/home/seed/work/source# ./test
..............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..............
root@VM:/home/seed/work/source# ./rmdev.sh
ls: cannot access '/dev/mydev': No such file or directory
root@VM:/home/seed/work/source# make clean
make -C /lib/modules/`uname -r`/build M=`pwd` clean
```

Kernel log:



```
root@VM: /home/seed/work/source
root@VM: /home/seed/work/source 80x30
dmesg | grep OS_AS5
[ 4483.251978] OS_AS5:init_modules():..............Sta
rt..............
[ 4483.251980] OS_AS5:init_modules(): register chrdev(2
45,0)
[ 4483.251981] OS_AS5:init_modules():allocate dma buffe
r
[ 4605.145471] OS_AS5:drv_open(): device open
[ 4605.145482] OS_AS5:drv_ioctl:My STUID is 118020362
[ 4605.145485] OS_AS5:drv_ioctl():RW OK
[ 4605.145488] OS_AS5:drv_ioctl():IOC OK
[ 4608.598174] OS_AS5:drv_ioctl():Blocking IO
[ 4608.598178] OS_AS5:drv_write(): queue work
[ 4608.598179] OS_AS5:drv_write(): block
[ 4611.220343] OS_AS5:drv_arithmetic_routine():100 p 20
000 = 225077
[ 4611.220462] OS_AS5:drv_read():ans is 225077
[ 4611.220495] OS_AS5:drv_ioctl():NonBlocking IO
[ 4611.220497] OS_AS5,drv_write(): queue work
[ 4611.220501] OS_AS5:drv_ioctl():Wait Readable 1
[ 4613.793020] OS_AS5:drv_arithmetic_routine():100 p 20
000 = 225077
[ 4614.296813] OS_AS5:drv_read():ans is 225077
[ 4614.297303] OS_AS5:drv_release(): device close
[ 4770.913777] OS_AS5:exit_modules():free dma buffer
[ 4770.913779] OS_AS5:exit_modules():unregister chrdev
[ 4770.913782] OS_AS5:exit_modules():.............End.
.............
root@VM:/home/seed/work/source#
```

What did I learn from this assignment?

I learned how interrupt I/O works.

I learned that how to manipulate I/O using ioctl() function, linux sees every device as a file node, access a device is like reading and writing a file.

I learned how to transfer data between kernel space and user space.

I learned how Direct Memory Access buffer works.

I learned the difference between blocking and non-blocking I/O, the main difference is that in non-blocking mode, when user wants to access the data, he/she needs to first check whether DMA buffer is readable.

I learned what is work routine function, and how to queue and flush it.