微算機系統實習

實驗5

日期:2021/05/21

1. 組員姓名：

第一組

資工二　108590044　何柏憲

資工二　108590452　林峻霆

1. 實驗步驟截圖與說明：

實驗目標:

• 創建file\_operations資料結構   
open release read write I/O

• 掛載與卸載驅動程式  
• 控制裝置驅動  
• 讀取裝置驅動狀態  
• 在TX2執行

實驗過程:

1.首先我們先參考範例了解此次實驗要做什麼，發現是延伸lab5 和lab2 的實驗

2.因為這次是遠距教學，第一週我們組員有見面一起討論此實驗，第二週則是線上meet

3.第二週從了解助教的程式碼後實作，在修改成此次實驗要求的程式碼。我們在demsg 的時候一直出錯後來發現是一直沒有write進去。

4. 讀取參數並執行

5.錄影並執行課程之要求

以下是我們的程式碼：

Makefile:

obj-m := fuck.o

kernel\_DIR := /usr/src/linux-headers-4.9.201-tegra-ubuntu18.04\_aarch64/kernel-4.9/

PWD := $(shell pwd)

all:

make -C $(kernel\_DIR) SUBDIRS=$(PWD)

clean:

rm \*.o \*.ko \*.mod.c

.PHONY:

clean

gpio.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <fcntl.h>

#include <map>

#include <string>

#include <iostream>

using namespace std;

void setGPIO(unsigned int gpio, string status)

{

int io;

io=open("/dev/demo", O\_WRONLY);//對/dev/demo進行唯寫

if (io<0){

perror("gpio error");//開檔失敗

return;

}

char buf[10] = {'0'};

if (status == "on") {

strcpy(buf,(to\_string(gpio)+ " 1").c\_str());//gpio跟status放入buf中

}

else {

strcpy(buf, (to\_string(gpio) + " 0").c\_str());

}

cout << buf << endl;

write(io, buf,8);//寫入

close(io);//關檔

return;

}

int main(int argc,char \*argv[]){

char buf[1024] = {0};

string led = argv[1];

int arr\_pin[] = {396,397,255,481};

int pin = arr\_pin[led[0]-'1'];

if(argc==3){

setGPIO(pin,argv[2]);

}

if(argc==2){

FILE \*fp = fopen("/dev/demo","w+");

sprintf(buf,"%d",pin);

fwrite(buf,sizeof(pin),1,fp);

fread(buf,sizeof(buf),1,fp);

printf("LED%s Status: %s\n",argv[1],buf);

fclose(fp);

}

return 0;

}

fuck..cpp

#include <linux/init.h>

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/fs.h>

#include <asm/uaccess.h>

#define MAJOR\_NUM 60

#define MODULE\_NAME "demo"

static int iCount = 0;

static char status[4]={'-1','-1','-1','-1'};

static ssize\_t drv\_read(struct file \*filp, char \*buf, size\_t count, loff\_t \*ppos) {

printk("device read\n");

char userChar[count];

copy\_from\_user(userChar, buf, count);

printk("Count: %d",count);

userChar[count - 1] = '\0';

printk("userChar(char): %s\n", userChar);

char gpio[10] = {0};

char fileStr[100];

char status[10] ={0};

strcpy(gpio,userChar);

struct file \*io;

loff\_t pos = 0;

mm\_segment\_t old\_fs;

old\_fs = get\_fs();

set\_fs(get\_ds());

printk("Read-Value: %s",gpio);

sprintf(fileStr,"/sys/class/gpio/gpio%s/value",gpio);

io = filp\_open(fileStr, O\_RDONLY, 0);

if(IS\_ERR(io)){

status[0]= '0';

copy\_to\_user(buf,status,sizeof(status));

return count;

}

vfs\_read(io, status, 1, &pos);

filp\_close(io, NULL);

pos = 0;

set\_fs(old\_fs);

copy\_to\_user(buf,status,sizeof(status));

return count;

}

static ssize\_t drv\_write(struct file \*filp, const char \*buf, size\_t count, loff\_t \*ppos) {

printk("device write\n");

printk("%d\n", iCount);

printk("W\_buf\_size: %d\n", (int)count);

char userChar[12] = {'0'};

copy\_from\_user(userChar, buf, count);

userChar[count - 1] = '\0';

printk("userChar(char): %s\n", userChar);

char gpio[10] = {'0'};

strncpy(gpio, userChar, 5);

printk("gpio: %s, length: %d\n", gpio, strlen(gpio));

struct file \*io;

loff\_t pos = 0;

mm\_segment\_t old\_fs;

old\_fs = get\_fs();

set\_fs(get\_ds());

io = filp\_open("/sys/class/gpio/export", O\_WRONLY, 0);

vfs\_write(io, gpio, strlen(gpio), &pos);

filp\_close(io, NULL);

//set-dir

if(strncmp(gpio, "396 1",5) == 0 || strncmp(gpio, "396 0",5) == 0) {

io = filp\_open("/sys/class/gpio/gpio396/direction", O\_WRONLY, 0);

printk("out");

vfs\_write(io, "out", 3, &pos);

filp\_close(io, NULL);

}

pos = 0;

//set-value

if(strncmp(gpio, "396 1",5) == 0 || strncmp(gpio, "396 0",5) == 0) {

io = filp\_open("/sys/class/gpio/gpio396/value", O\_WRONLY, 0);

vfs\_write(io, &userChar[count -4], 1, &pos);

filp\_close(io, NULL);

}

if(strncmp(gpio, "481 0",5) == 0 || strncmp(gpio, "481 1",5) == 0) {

io = filp\_open("/sys/class/gpio/gpio481/direction", O\_WRONLY, 0);

vfs\_write(io, "out", 3, &pos);

filp\_close(io, NULL);

}

pos = 0;

//set-value

if(strncmp(gpio, "481 0",5) == 0 || strncmp(gpio, "481 1",5) == 0 ){

io = filp\_open("/sys/class/gpio/gpio481/value", O\_WRONLY, 0);

vfs\_write(io, &userChar[count -4], 1, &pos);

filp\_close(io, NULL);

}

if(strncmp(gpio, "255 0",5) == 0 || strncmp(gpio, "255 1",5) == 0) {

io = filp\_open("/sys/class/gpio/gpio255/direction", O\_WRONLY, 0);

vfs\_write(io, "out", 3, &pos);

filp\_close(io, NULL);

}

pos = 0;

//set-value

if(strncmp(gpio, "255 0",5) == 0 || strncmp(gpio, "255 1",5) == 0) {

io = filp\_open("/sys/class/gpio/gpio255/value", O\_WRONLY, 0);

vfs\_write(io, &userChar[count -4], 1, &pos);

filp\_close(io, NULL);

}

if(strncmp(gpio, "397 1",5) == 0 || strncmp(gpio, "397 0",5) == 0) {

io = filp\_open("/sys/class/gpio/gpio397/direction", O\_WRONLY, 0);

vfs\_write(io, "out", 3, &pos);

filp\_close(io, NULL);

}

pos = 0;

//set-value

if(strncmp(gpio, "397 1",5) == 0|| strncmp(gpio, "397 0",5) == 0){

io = filp\_open("/sys/class/gpio/gpio397/value", O\_WRONLY, 0);

vfs\_write(io, &userChar[count -4], 1, &pos);

filp\_close(io, NULL);

}

pos = 0;

set\_fs(old\_fs);

iCount++;

return count;

}

long drv\_ioctl(struct file \*flip, unsigned int cmd, unsigned long arg) {

printk("device ioctl\n");

return 0;

}

static int drv\_open(struct inode \*inode, struct file \*filp){

printk("device open\n");

return 0;

}

static int drv\_release(struct inode \*inode, struct file \*filp){

printk("device close\n");

return 0;

}

struct file\_operations drv\_fops = {

read : drv\_read,

write : drv\_write,

unlocked\_ioctl : drv\_ioctl,

open : drv\_open,

release : drv\_release,

};

static int demo\_init(void) {

if(register\_chrdev(MAJOR\_NUM, "demo", &drv\_fops) < 0) {

printk("<1>%s : can't get major %d\n", MODULE\_NAME, MAJOR\_NUM);

return (-EBUSY);

}

printk("<1>%s : started\n", MODULE\_NAME);

return 0;

}

static void demo\_exit(void) {

unregister\_chrdev(MAJOR\_NUM, "demo");

printk("<1>%s : removed\n", MODULE\_NAME);

}

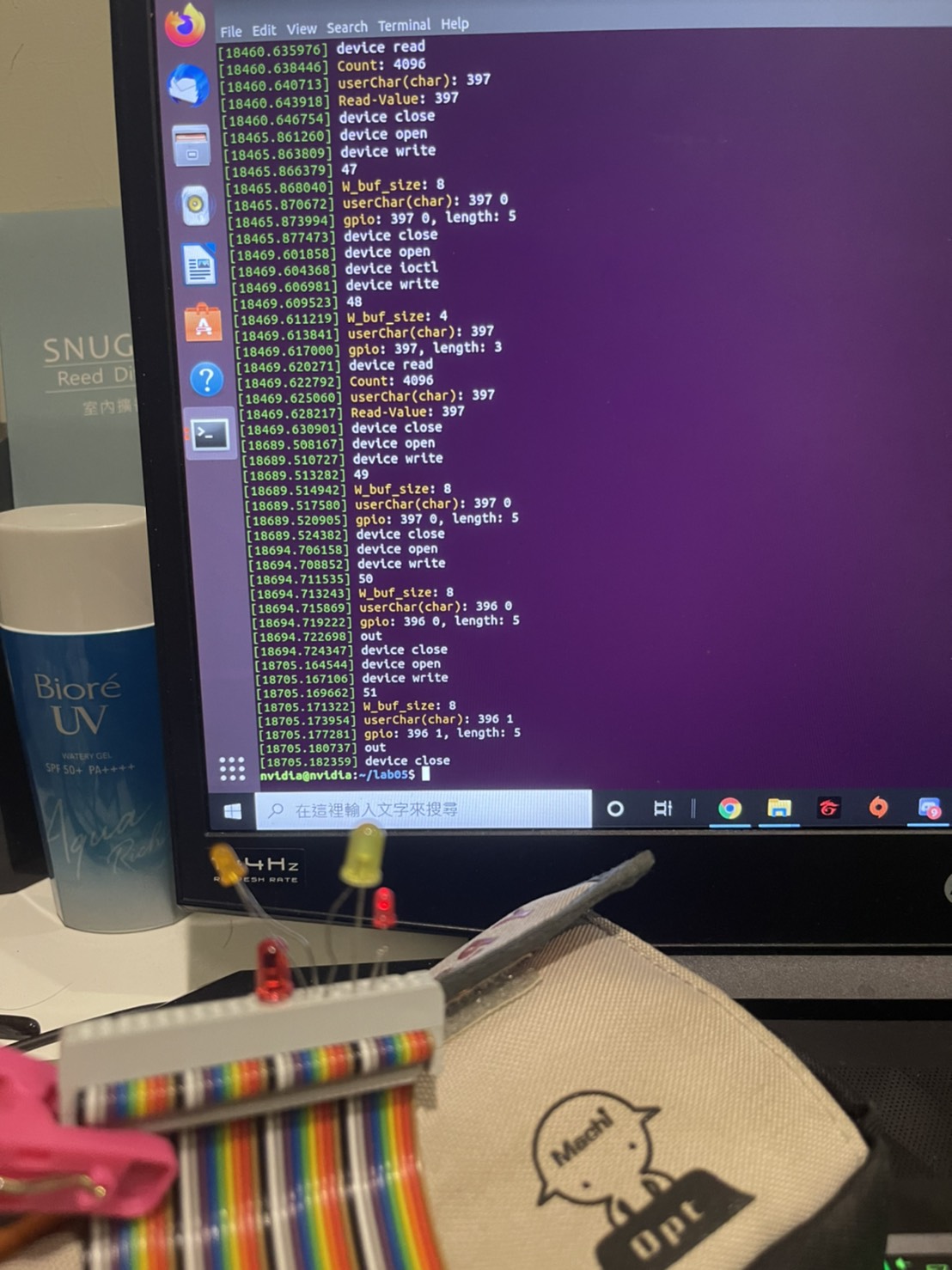
module\_init(demo\_init);

module\_exit(demo\_exit);

MODULE\_LICENSE("GPL");

以下是實驗結果

<https://www.youtube.com/watch?v=371fRE7DmZk>

demsg的結果和Led的結果：

影片連結：

1. 組員貢獻比例：

何柏憲：50%

林峻霆：50%

1. 心得：

何柏憲：

　　 這次實驗跟之前的內容差不多，把程式碼整合起來就和撰寫新的驅動程式檔案，不過這次都是要在家裡自己做，少了助教的當場協助，做起來也花了更多的時間，第一週去組員來到我家的時候超害怕的，那時疫情真的恐怖，但我們經過一番討論後，進度也做了起來，第二週再延續前一皺的進度，我們比較順利的完成最後的目的，也終於可以好好的去吃飯了。

林峻霆：

這次實驗延續了前兩次( lab2 和 lab5一開始 )的實驗內容，我覺得遠距的微算機實習真的頗難的，第一週去組員家的時候超害怕的，因為ＴＸ2在我這裡所以非得跑一趟（電腦是mac），回程的時候還下大雨，至少最後在第一週進度不多後，助教給了我們不少的提示，雖然第二週還是做到爆肝，但至少順利的完成了，也搞懂了原理。希望下次能夠早點完成才能好好吃飯（肚子好餓啊）。