微算機系統實習

第 13 組專案報告

LAB 07

實作 Char Device Driver

組別： 13

班級、姓名與學號：

醫工三 B812110004 葉芸茜

醫工三 B812110011 湯青秀

日期：2024.06.05

一、實驗內容：

1. lab 7-1

練習撰寫一個可掛載於系統的驅動模組

(1) 項目一：製作簡易驅動程式(30%)

分別在虛擬機和 TX2 上成功執行

需查看系統訊息

(2) 項目二 :製作字元驅動程式(40%)

在虛擬機上製作字元驅動程式

需查看系統訊息

2. lab 7-2

練習撰寫一個可掛載於系統的字元驅動

用 C/C++設計另一個程式來控制掛載中的字元驅動

能利用字元驅動來控制 GPIO

使用 Linux Kernel 的檔案存取方式來處理 LED 控制

請編寫一個字元裝置(character device)驅動程式

使用 mknod 指令在/dev/下建立一個名為 demo 的 node

編寫字元驅動需使用 module_init 及 module_exit

模組須使用 Makefile 編譯

(1) 項目一：創建 file_operations 資料結構 (15%)

每個 function 需印出以下規定訊息

Open: Enter Open function

Release: Enter Release function

Read: Enter Read function

Write: Enter Write function

I/O Control: Enter I/O Control function

(2) 項目二：掛載與卸載驅動程式(15%)

初始(掛載)使用 register_chrdev function

離開(卸載)使用 unregister_chrdev function

(3) 項目三：控制裝置驅動 (20%)

必須能讀取命令參數，使指定的 LED 燈點亮或熄滅(on/off)

能分別點亮及熄滅 4 顆 LED 燈，此動作需於驅動程式中完成

Ex: ./Lab7 LED1 on (第一顆燈狀態為亮，其他 LED 狀態不變)

(4) 項目四：讀取裝置驅動狀態 (20%)

必須在終端機上顯示被點亮的燈號

Ex: ./Lab7 LED1 (讀取 LED1 點燈狀態)

顯示“LED1 Status: 0”(Terminal 回傳 LED1 狀態為 0，表示未亮)

程式於 TX2 嵌入式平台上執行

二、實驗過程及結果：

1. 實驗過程

(1) lab 7-1

(a) 項目一

(i) hellod.c

```c
#include <linux/kernel.h>
#include <linux/module.h>

static int __init tx2_hello_module_init(void){
    printk("Hello, TX2 module is installed !\n");
    return 0;
}

static void __exit tx2_hello_module_cleanup(void){
    printk("Good-bye, TX2 module was removed!\n");
}

module_init(tx2_hello_module_init);
module_exit(tx2_hello_module_cleanup);
MODULE_LICENSE("GPL");
```

(ii) Makefile

```
obj-m := hellod.o

kernel_DIR := /usr/src/linux-headers-5.4.0-150-generic

PWD := $(shell pwd)

all:
    make -C $(kernel_DIR) M=$(PWD)
clean:
    rm *.o *.ko *.mod.c
.PHONY:
    clean
```

其中 kernel_DIR 在虛擬機為
/usr/src/linux-headers-5.4.0-150-generic
在 TX2 則為
/usr/src/linux-headers-4.9.201-tegra-ubuntu18.04_aarch64/kernel-4.9/

（b）項目二

（i） demo.c

實作讀寫等函數

```c
static ssize_t drv_write(struct file *flip, 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);
    raw_copy_from_user(userChar, buf, count);
    userChar[count - 1] = 0;
    printk("userChar: %s\n", userChar);
    printk("userChar: %d\n", (int)sizeof(userChar));
    iCount++;
    return count;
}

```

```c
long drv_ioctl(struct file *filp, 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;
}

static ssize_t drv_read(struct file *flip, char *buf, size_t count, loff_t *ppos){
    printk("device read\n");
    return count;
}
```

```c
struct file_operations drv_fops =
{
    read: drv_read,
    write: drv_write,
    unlocked_ioctl: drv_ioctl,
    open: drv_open,
    release: drv_release,
};
```

實作初始化以及退出函數

```c
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);
}
```

（ii） test.c

控制掛載好的 demo

```c
#include <stdio.h>
int main(){
    char buf[1024] = "Data Input 123456 hello world";
    FILE *fp = fopen("/dev/demo", "w+");
    if(fp == NULL){
        printf("can't open device!\n");
        return 0;
    }
    fwrite(buf, sizeof(buf), 1, fp);
    fread(buf, sizeof(buf), 1, fp);
    fclose(fp);
    return 0;
}
```

（2）lab 7-2

（a）demo.c

實作讀寫函數

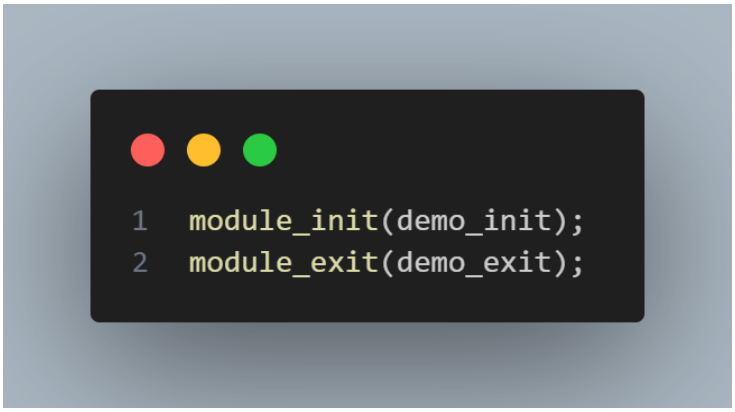
```c
static int driver_open(struct inode *inode, struct file *filp) {
    printk("Open: Enter Open function\n");
    return 0;
}

static int driver_close(struct inode *inode, struct file *filp) {
    printk("Release: Enter Release function\n");
    return 0;
}

long driver_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
{
    printk("I/O Control: Enter I/O Control function\n");
    return 0;
}

static ssize_t driver_read(struct file *filp, char *buf, size_t size, loff_t *f_pos) {
    printk("Read: Enter Release function\n");
    return 0;
}
```

```c
static ssize_t driver_write(struct file *filp, const char *buf, size_t size, loff_t *f_pos) {
    printk("Write: Enter Write function\n");
    if(copy_from_user(userChar, buf, size) == 0){
        userChar[size - 1] = '\0';
        int ledIndex;
        int pin;

        printk("%s\n", userChar);
        if(userChar[0] == 'g') {
            ledIndex = userChar[4] - '1';
            printk("LED%d Status: %d\n", ledIndex + 1, ledBook[ledIndex]);
        }
        else if (userChar[1] == 'n') {
            ledIndex = userChar[3] - '1';
            pin = gpioPin[ledIndex];
            gpio_set_value(pin, 1);
            ledBook[ledIndex] = 1;
        }
        else if (userChar[1] == 'f') {
            ledIndex = userChar[4] - '1';
            pin = gpioPin[ledIndex];
            gpio_set_value(pin, 0);
            ledBook[ledIndex] = 0;
        }
        else {
            printk("Error: command not found");
        }
    } else{
        printk("Error: Write Error\n");
    }
    return size;
}
```

實作初始化及退出函數

```c
static int demo_init(void) {
    int result;
    printk("<1>demo: started\n");

    char* ledName[4] = { "LED1", "LED2", "LED3", "LED4" };
    int i;
    for (i = 0; i < 4; i++) {
        if (gpio_is_valid(gpioPin[i]) == 0) {
            printk("pin %d is no valid\n", gpioPin[i]);
            return -EBUSY;
        }

        if (gpio_request(gpioPin[i], ledName[i]) < 0) {
            printk("pin %d is busy\n", gpioPin[i]);
            return -EBUSY;
        }

        gpio_direction_output(gpioPin[i], 0);
        gpio_export(gpioPin[i], false);
    }

    result = register_chrdev(DRIVER_MAJOR, DRIVER_NAME, &driver_fops);
    if (result < 0) {
        printk("<1>demo: Failed to register character device\n");
        return result;
    }

    return 0;
}
```

```c
static void demo_exit(void) {
    printk("<1>demo: removed\n");
    int i;
    for (i = 0; i < 4; i++) {
        gpio_free(gpioPin[i]);
    }

    /* Unregister character device */
    unregister_chrdev(DRIVER_MAJOR, DRIVER_NAME);
}
```



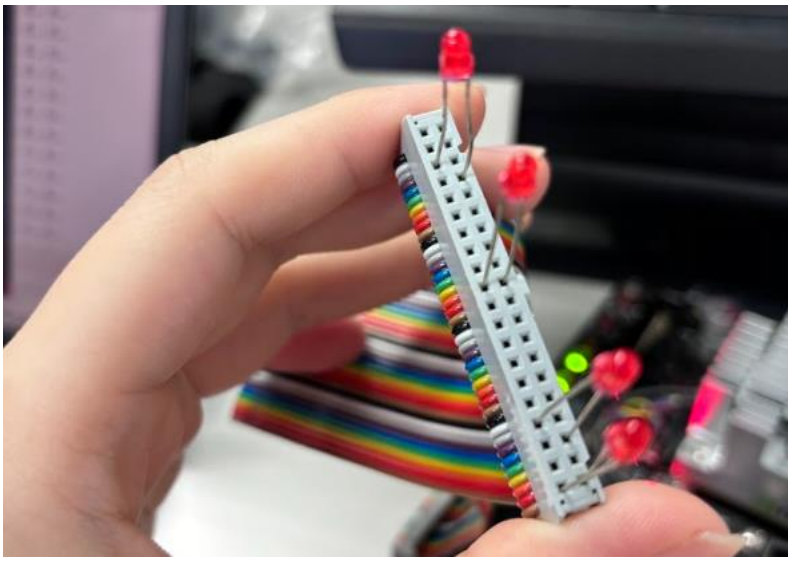
```
module_init(demo_init);
module_exit(demo_exit);
```

（b）led.cpp

控制 demo

```cpp
#include <iostream>
#include <cstdio>

using namespace std;

int main(int argc, char *argv[]) {
    char buffer[100] = { 0 };
    FILE *fp = fopen("/dev/demo", "w+");

    if (fp != NULL) {
        if (argc == 3) {
            snprintf(buffer, sizeof(buffer), "%s %c", argv[2], argv[1][3]);
            fwrite(buffer, sizeof(buffer), 1, fp);
        } else if (argc == 2) {
            snprintf(buffer, sizeof(buffer), "get %c", argv[1][3]);
        } else {
            cout << "Error: number of arg\n";
            cout << "number of arg: " << argc << endl;
        }

        fwrite(buffer, sizeof(buffer), 1, fp);
    } else {
         cout << "Error: Failed to open file\n";
         return 0;
    }

    fread(buffer, sizeof(buffer), 1, fp);
    fclose(fp);
    return 0;
}
```

LED 腳位



LED1： 7(VCC)、 6(GND) -> GPIO396
LED2：19(VCC)、20(GND) -> GPIO429
LED3：35(VCC)、34(GND) -> GPIO395
LED4：40(VCC)、39(VCC) -> GPIO393

2. 預期實驗結果

能成功掛載.ko 並在輸入指令後用 dmesg 查看到對應系統訊息。在 lab7-2 中，輸入./led LEDx on 可使 LEDx 亮燈，./led LEDx off 可使 LEDx 變暗，而./led LEDx 則會於系統訊息顯示 LEDx 目前的狀態，如亮燈則 state 為 1，暗燈時 state 為 0

3. 實際上的結果

(1) lab 7-1

(a) 項目一

(i) 虛擬機

掛載/卸除

```
  LD [M]  /home/nvidia/Desktop/lab07/hellod.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-150-generic'
nvidia@ubuntu:~/Desktop/lab07$ sudo insmod hellod.ko
nvidia@ubuntu:~/Desktop/lab07$ sudo rmmod hellod.ko
nvidia@ubuntu:~/Desktop/lab07$ dmesg
```

系統訊息

```
[ 1683.941673] Hello, TX2 module is installed !
[ 1738.879043] e1000: ens33 NIC Link is Down
[ 1744.932923] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Fl
[ 1842.874089] Good-bye, TX2 module was removed!
nvidia@ubuntu:~/Desktop/lab07$
```

（ii） TX2

掛載/卸除

```
> ^C
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo insmod hellod.ko
[sudo] password for nvidia:
nvidia@nvidia-desktop:~/Desktop/lab07$ dmesg
[10403.260428] hellod: loading out-of-tree module taints kernel.
[10403.266774] Hello, TX2 module is installed !
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo rmmod hellod.ko
nvidia@nvidia-desktop:~/Desktop/lab07$ dmesg
```

系統訊息

```
[ 1664.763341] Hello, TX2 module is installed !
[ 1673.172026] Good-bye, TX2 module was removed!
```

查看掛載狀態

```
nvidia@ubuntu:~/Desktop/lab07$ sudo insmod hellod.ko
nvidia@ubuntu:~/Desktop/lab07$ lsmod | grep hellod
hellod                 16384  0
nvidia@ubuntu:~/Desktop/lab07$ sudo rmmod hellod.ko
nvidia@ubuntu:~/Desktop/lab07$ dmesg
```

（b）項目二

make & insmod

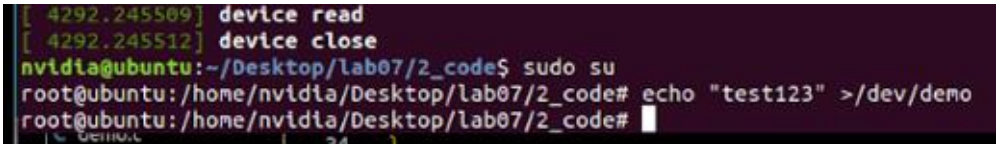
```
make: *** [all] Error 2
nvidia@ubuntu:~/Desktop/lab07/2_code$ make
make -C /usr/src/linux-headers-5.4.0-150-generic M=/home/nvidia/Desktop/lab07/2_
code
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-150-generic'
  CC [M]  /home/nvidia/Desktop/lab07/2_code/demo.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/nvidia/Desktop/lab07/2_code/
demo.o
see include/linux/module.h for more information
  CC [M]  /home/nvidia/Desktop/lab07/2_code/demo.mod.o
  LD [M]  /home/nvidia/Desktop/lab07/2_code/demo.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-150-generic'
nvidia@ubuntu:~/Desktop/lab07/2_code$ sudo insmod demo.ko
nvidia@ubuntu:~/Desktop/lab07/2_code$
```

mknod

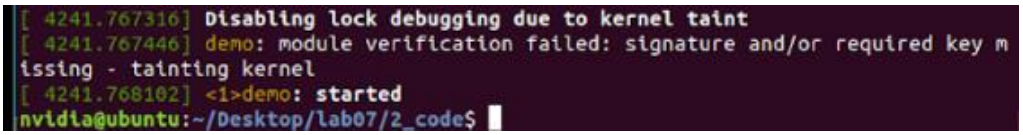
```
nvidia@ubuntu:~$ sudo mknod /dev/demo c 60 0
[sudo] password for nvidia:
nvidia@ubuntu:~$ cd /dev/
nvidia@ubuntu:/dev$ ls -l demo
crw-r--r-- 1 root root 60, 0 May 20 00:26 demo
nvidia@ubuntu:/dev$
```

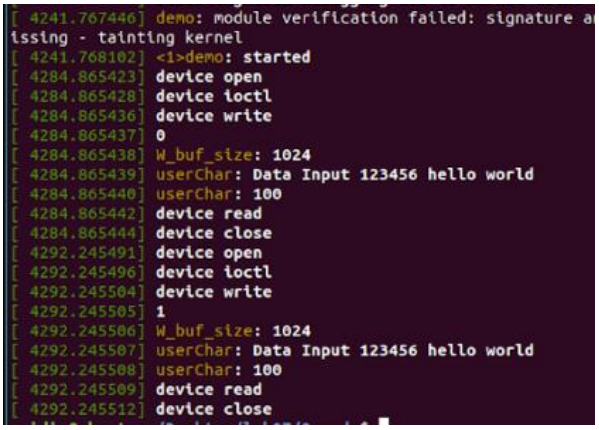
呼叫 test.o 執行

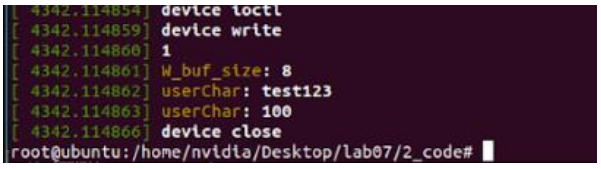
```
issing - tainting kernel
[ 4241.768102] <1>demo: started
nvidia@ubuntu:~/Desktop/lab07/2_code$ sudo ./test.o
nvidia@ubuntu:~/Desktop/lab07/2_code$ sudo ./test.o
nvidia@ubuntu:~/Desktop/lab07/2_code$ dmesg
```



系統訊息



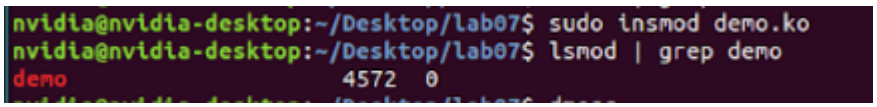


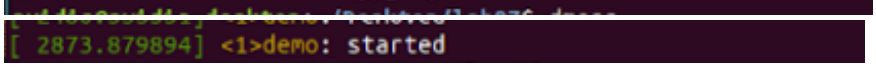


(2) lab 7-2

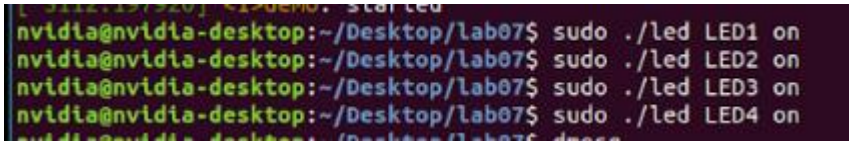
操作影片：lab7_2.MOV

(a) 掛載 demo.ko，查看系統訊息會顯示 started





(b) 控制 LED 亮燈



```
[ 3132.213324] Open: Enter Open function
[ 3132.217011] I/O Control: Enter I/O Control function
[ 3132.222106] Write: Enter Write function
[ 3132.226296] on 1
[ 3132.228352] Read: Enter Release function
[ 3132.232511] Release: Enter Release function
[ 3137.488616] Open: Enter Open function
[ 3137.492351] I/O Control: Enter I/O Control function
[ 3137.497366] Write: Enter Write function
[ 3137.501454] on 2
[ 3137.503497] Read: Enter Release function
[ 3137.507476] Release: Enter Release function
[ 3142.220897] Open: Enter Open function
[ 3142.224621] I/O Control: Enter I/O Control function
[ 3142.229597] Write: Enter Write function
[ 3142.233450] on 3
[ 3142.235289] Read: Enter Release function
[ 3142.239263] Release: Enter Release function
[ 3145.777080] Open: Enter Open function
[ 3145.780812] I/O Control: Enter I/O Control function
[ 3145.785717] Write: Enter Write function
[ 3145.789569] on 4
[ 3145.791408] Read: Enter Release function
[ 3145.795345] Release: Enter Release function
```

（c）查看 LED 狀態

```
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo ./led LED4
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo ./led LED3
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo ./led LED2
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo ./led LED1
nvidia@nvidia-desktop:~/Desktop/lab07$ dmesg
[ 3145.795345] Release: Enter Release function
[ 3156.998470] Open: Enter Open function
[ 3157.002232] I/O Control: Enter I/O Control function
[ 3157.007208] Write: Enter Write function
[ 3157.011097] get 4
[ 3157.013021] LED4 Status: 1
[ 3157.015750] Read: Enter Release function
[ 3157.019706] Release: Enter Release function
[ 3161.832056] Open: Enter Open function
[ 3161.835810] I/O Control: Enter I/O Control function
[ 3161.840718] Write: Enter Write function
[ 3161.844569] get 3
[ 3161.846508] LED3 Status: 0
[ 3161.849244] Read: Enter Release function
[ 3161.853187] Release: Enter Release function
[ 3165.851621] Open: Enter Open function
[ 3165.855349] I/O Control: Enter I/O Control function
[ 3165.860268] Write: Enter Write function
[ 3165.864182] get 2
[ 3165.866127] LED2 Status: 0
[ 3165.868830] Read: Enter Release function
[ 3165.872767] Release: Enter Release function
[ 3167.462251] Open: Enter Open function
[ 3167.465949] I/O Control: Enter I/O Control function
[ 3167.470857] Write: Enter Write function
[ 3167.474705] get 1
[ 3167.476627] LED1 Status: 0
[ 3167.479345] Read: Enter Release function
[ 3167.483278] Release: Enter Release function
```

(d)關閉 LED 燈

```
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo ./led LED1 off
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo ./led LED2 off
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo ./led LED3 off
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo ./led LED4 off
nvidia@nvidia-desktop:~/Desktop/lab07$ dmesg
[ 3185.088180] Open: Enter Open function
[ 3185.091899] I/O Control: Enter I/O Control function
[ 3185.096803] Write: Enter Write function
[ 3185.100651] off 1
[ 3185.102594] Read: Enter Release function
[ 3185.106532] Release: Enter Release function
[ 3189.515356] Open: Enter Open function
[ 3189.519212] I/O Control: Enter I/O Control function
[ 3189.524906] Write: Enter Write function
[ 3189.528823] off 2
[ 3189.530787] Read: Enter Release function
[ 3189.534736] Release: Enter Release function
[ 3193.672630] Open: Enter Open function
[ 3193.676371] I/O Control: Enter I/O Control function
[ 3193.681346] Write: Enter Write function
[ 3193.685277] off 3
[ 3193.687259] Read: Enter Release function
[ 3193.691522] Release: Enter Release function
[ 3196.689252] Open: Enter Open function
[ 3196.692939] I/O Control: Enter I/O Control function
[ 3196.697872] Write: Enter Write function
[ 3196.701747] off 4
[ 3196.703678] Read: Enter Release function
[ 3196.707615] Release: Enter Release function
```

(e) 移除掛載

```
nvidia@nvidia-desktop:~/Desktop/lab07$ sudo rmmod demo
nvidia@nvidia-desktop:~/Desktop/lab07$ dmesg
[ 3235.599815] <1>demo: removed
nvidia@nvidia-desktop:~/Desktop/lab07$ lsmod | grep demo
```

4. 遇到的問題&問題怎麼解決

這次主要遇到的問題是 lab7_2 時 LED 燈不會亮，我們使用了以前的程式碼去測試是否能正常開啟燈的狀態，發現是可行的，但使用掛載後卻無法操控對應 gpio。後來我們到/dev 目錄檢查，才發現在該目錄底下並沒有名稱為 demo 的設備文件，而創立後設備文件的權限也無法進行寫入，所以我們又另外設置權限，最終才可以順利操控 LED 燈。

在\dev 下以 ls -l 查看，發現尚未有 demo

```
crw-------  1 root root     10,  46 五  27 18:32 cpu_freq_min
crw-------  1 root root     10, 203 一  28  2018 cuse
drwxr-xr-x  8 root root         160 五  27 18:32 disk
crw-------  1 root root     10,  44 五  27 18:32 emc_freq_min
```

創建 demo 的 node(沒有權限)

```
crw-------  1 root root     10, 203  一  28  2018 cuse
crw-r--r--  1 root root     60,   0  五  27 19:35 demo
drwxr-xr-x  8 root root          160  五  27 18:32 disk
crw-------  1 root root     10,  44  五  27 18:32 emc_freq_min
```

給予 demo 權限

```
nvidia@nvidia-desktop:/dev$ sudo chmod 666 /dev/demo
nvidia@nvidia-desktop:/dev$ ls -l
total 0
crw-------  1 root root     10, 235  五  27 18:32 autofs
drwxr-xr-x  2 root root         1340  五  27 18:32 block
crw-------  1 root root     10, 234  一  28  2018 btrfs-control
drwxr-xr-x  3 root root           60  一   1  1970 bus
crw-rw----  1 root video    243,   1  五  27 18:32 camchar-dbg
crw-rw----  1 root video    243,   0  五  27 18:32 camchar-echo
crw-rw----  1 root video    250,   0  五  27 18:32 capture-vi-channel0
crw-rw----  1 root video    250,   1  五  27 18:32 capture-vi-channel1
crw-rw----  1 root video    250,  10  五  27 18:32 capture-vi-channel10
crw-rw----  1 root video    250,  11  五  27 18:32 capture-vi-channel11
crw-rw----  1 root video    250,  12  五  27 18:32 capture-vi-channel12
crw-rw----  1 root video    250,  13  五  27 18:32 capture-vi-channel13
crw-rw----  1 root video    250,  14  五  27 18:32 capture-vi-channel14
crw-rw----  1 root video    250,   2  五  27 18:32 capture-vi-channel2
crw-rw----  1 root video    250,   3  五  27 18:32 capture-vi-channel3
crw-rw----  1 root video    250,   4  五  27 18:32 capture-vi-channel4
crw-rw----  1 root video    250,   5  五  27 18:32 capture-vi-channel5
crw-rw----  1 root video    250,   6  五  27 18:32 capture-vi-channel6
crw-rw----  1 root video    250,   7  五  27 18:32 capture-vi-channel7
crw-rw----  1 root video    250,   8  五  27 18:32 capture-vi-channel8
crw-rw----  1 root video    250,   9  五  27 18:32 capture-vi-channel9
drwxr-xr-x  2 root root         6060  五  27 18:32 char
crw-------  1 root root      5,   1  五  27 18:32 console
crw-------  1 root root     10,  41  五  27 18:32 constraint_cpu_freq
crw-------  1 root root     10,  40  五  27 18:32 constraint_gpu_freq
crw-------  1 root root     10,  39  五  27 18:32 constraint_online_cpus
crw-------  1 root root     10,  54  五  27 18:32 cpu_dma_latency
crw-------  1 root root     10,  45  五  27 18:32 cpu_freq_max
crw-------  1 root root     10,  46  五  27 18:32 cpu_freq_min
crw-------  1 root root     10, 203  一  28  2018 cuse
crw-rw-rw-  1 root root     60,   0  五  27 19:35 demo
drwxr-xr-x  8 root root          160  五  27 18:32 disk
```

三、程式碼

1. lab 7-1

（1）項目一

hellod.c

```
#include <linux/kernel.h>
#include <linux/module.h>

static int __init tx2_hello_module_init(void){
    printk("Hello, TX2 module is installed !\n");
    return 0;
}

static void __exit tx2_hello_module_cleanup(void){
    printk("Good-bye, TX2 module was removed!\n");
}
```

```
module_init(tx2_hello_module_init);
module_exit(tx2_hello_module_cleanup);
MODULE_LICENSE("GPL");
```

Makefile（ubuntu）

```
obj-m := hellod.o

kernel_DIR := /usr/src/linux-headers-5.4.0-150-generic

PWD := $(shell pwd)

all:
      make -C $(kernel_DIR) M=$(PWD)
clean:
      rm *.o *.ko *.mod.c
.PHONY:
      clean
```

Makefile（tx2）

```
obj-m := hellod.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
```

（2）项目二

demo.c

```
#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 userChar[100];


static ssize_t drv_write(struct file *flip, 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);
    raw_copy_from_user(userChar, buf, count);
    userChar[count - 1] = 0;
    printk("userChar: %s\n", userChar);
    printk("userChar: %d\n", (int)sizeof(userChar));
    iCount++;
    return count;
}

long drv_ioctl(struct file *filp, 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;
}

static ssize_t drv_read(struct file *flip, char *buf, size_t
count, loff_t *ppos){
    printk("device read\n");
    return count;
}

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);
```

test.c

```
#include <stdio.h>
int main(){
    char buf[1024] = "Data Input 123456 hello world";
    FILE *fp = fopen("/dev/demo", "w+");
    if(fp == NULL){
        printf("can't open device!\n");
        return 0;
    }
    fwrite(buf, sizeof(buf), 1, fp);
    fread(buf, sizeof(buf), 1, fp);
    fclose(fp);
    return 0;
}
```

Makefile (ubuntu)

```
obj-m := hellod.o

kernel_DIR := /usr/src/linux-headers-5.4.0-150-generic

PWD := $(shell pwd)

all:
      make -C $(kernel_DIR) M=$(PWD)
clean:
      rm *.o *.ko *.mod.c
.PHONY:
```

```
clean
```

2. lab 7-2

demo.c

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/gpio.h>
#include <asm/uaccess.h>

MODULE_LICENSE("Dual BSD/GPL");

#define DRIVER_MAJOR 60
#define DRIVER_NAME "demo"

int gpioPin[4] = { 396, 429, 395, 393};
int ledPin[4] = { 7, 12, 18, 37 };
int ledBook[4] = { 0, 0, 0, 0 };
static char userChar[100];

static int driver_open(struct inode *inode, struct file *filp) {
    printk("Open: Enter Open function\n");
    return 0;
}

static int driver_close(struct inode *inode, struct file *filp)
{
    printk("Release: Enter Release function\n");
    return 0;
}

long driver_ioctl(struct file *filp, unsigned int cmd, unsigned
long arg)
{
    printk("I/O Control: Enter I/O Control function\n");
    return 0;
}

static ssize_t driver_read(struct file *filp, char *buf, size_t
size, loff_t *f_pos) {
    printk("Read: Enter Release function\n");
    return 0;
}

static ssize_t driver_write(struct file *filp, const char *buf,
size_t size, loff_t *f_pos) {
    printk("Write: Enter Write function\n");
```

```
    if(copy_from_user(userChar, buf, size) == 0){
        userChar[size - 1] = '\0';
        int ledIndex;
          int pin;

        printk("%s\n", userChar);
        if(userChar[0] == 'g') {
              ledIndex = userChar[4] - '1';
            printk("LED%d Status: %d\n", ledIndex + 1,
ledBook[ledIndex]);
        }
        else if (userChar[1] == 'n') {
              ledIndex = userChar[3] - '1';
              pin = gpioPin[ledIndex];
            gpio_set_value(pin, 1);
            ledBook[ledIndex] = 1;
        }
        else if (userChar[1] == 'f') {
              ledIndex = userChar[4] - '1';
              pin = gpioPin[ledIndex];
            gpio_set_value(pin, 0);
            ledBook[ledIndex] = 0;
        }
        else {
            printk("Error: command not found");
        }
    } else{
        printk("Error: Write Error\n");
    }
    return size;
}

static struct file_operations driver_fops = {
    .open = driver_open,
    .release = driver_close,
    .unlocked_ioctl = driver_ioctl,
    .read = driver_read,
    .write = driver_write,
};

static int demo_init(void) {
    int result;
    printk("<1>demo: started\n");

    char* ledName[4] = { "LED1", "LED2", "LED3", "LED4" };
    int i;
    for (i = 0; i < 4; i++) {
        if (gpio_is_valid(gpioPin[i]) == 0) {
            printk("pin %d is no valid\n", gpioPin[i]);
            return -EBUSY;
```

```
        }

        if (gpio_request(gpioPin[i], ledName[i]) < 0) {
            printk("pin %d is busy\n", gpioPin[i]);
            return -EBUSY;
        }

        gpio_direction_output(gpioPin[i], 0);
        gpio_export(gpioPin[i], false);
    }

    result = register_chrdev(DRIVER_MAJOR, DRIVER_NAME,
&driver_fops);
    if (result < 0) {
        printk("<1>demo: Failed to register character
device\n");
        return result;
    }

    return 0;
}

static void demo_exit(void) {
    printk("<1>demo: removed\n");
    int i;
    for (i = 0; i < 4; i++) {
        gpio_free(gpioPin[i]);
    }

    /* Unregister character device */
    unregister_chrdev(DRIVER_MAJOR, DRIVER_NAME);
}

module_init(demo_init);
module_exit(demo_exit);
```

led. cpp

```
#include <iostream>
#include <cstdio>

using namespace std;

int main(int argc, char *argv[]) {
    char buffer[100] = { 0 };
    FILE *fp = fopen("/dev/demo", "w+");

    if (fp != NULL) {
        if (argc == 3) {
```

```
            snprintf(buffer, sizeof(buffer), "%s %c", argv[2],
argv[1][3]);
            fwrite(buffer, sizeof(buffer), 1, fp);
        } else if (argc == 2) {
            snprintf(buffer, sizeof(buffer), "get %c",
argv[1][3]);
        } else {
            cout << "Error: number of arg\n";
            cout << "number of arg: " << argc << endl;
        }

        fwrite(buffer, sizeof(buffer), 1, fp);
    } else {
         cout << "Error: Failed to open file\n";
         return 0;
    }

    fread(buffer, sizeof(buffer), 1, fp);
    fclose(fp);
    return 0;
}
```

Makefile

```
obj-m :=demo.o

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

PWD := $(shell pwd)

all:
      g++ -o led led.cpp
      make -C $(kernel_DIR) SUBDIRS=$(PWD)
      sudo insmod demo.ko
clean:
      sudo rmmod demo.ko
      rm *.o *.ko *.mod.c
.PHONY:
      clean
```

四、本次實驗過程說明與解決方法:

1. 實驗過程

撰寫.c 及.cpp 檔及所需 Makefile → 使用 make 指令執行 Makefile → 在/dev/下新增 node →編寫字元裝置(建立 file_operations 資料結構)並編譯出 ko 檔(驅動模組檔案) → 將 ko 檔(驅動模組檔案)掛載並查看(dmesg)系統訊息→ 執行 LED 控制程式(呼叫對應驅動模組) → 控制特

定 LED 燈或讀取特定 LED 點燈狀態

2. 解決方法

LED 無法控制狀態，後來發現是/dev 目錄下無對應 demo，使用 mknod 建立 demo node 並給予它權限即可解決問題。

五、分工:

學號、組員	貢獻比例	工作內容
B812110004 葉芸茜	50%	文書處理、實驗設計與實作、程式規劃、測試與除錯
B812110011 湯青秀	50%	文書處理、實驗設計與實作、程式規劃、測試與除錯