Lab 4 簡易名字跑馬燈

I. Introduction

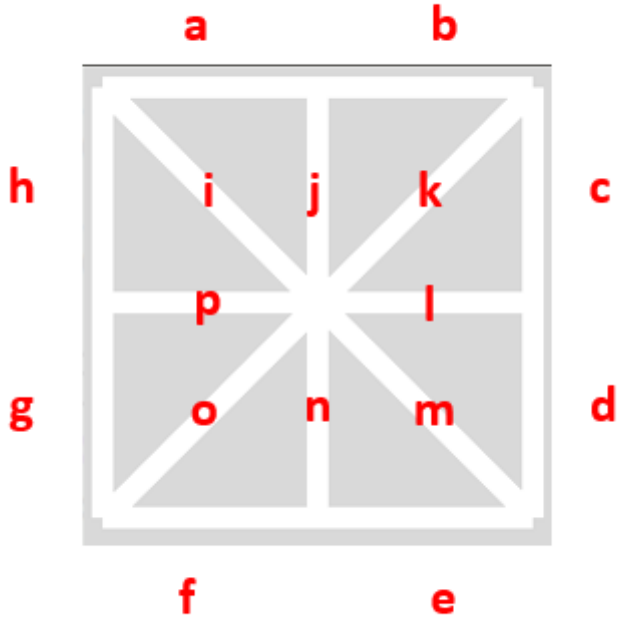
請在 ZC702 上撰寫一支名字跑馬燈。在 ZC702 上透過 writer 程式將英文字寫到 driver 當中，然後透過 reader 程式將該字從 driver 中讀出來，最後透過 socket 傳遞給 VM 上頭的 seg.py (http://seg.py) 程式，其會把該字用十六段顯示器 (GUI) 呈現出來。

II. Specification

- **driver (自行撰寫)**
  - 撰寫自行定義的 write function，將 writer 傳送過來的計數值給存起來。
  - 撰寫自行定義的 read function, 將經過處理之後的計數值回傳給使用者。
  - 所謂的處理，就是將字轉換為16段顯示器可以接受的資料格式 - **1 個長度為 16 的陣列，每一格儲存的不是 0 就是 1**。
  - 16段顯示器顯示資訊表

```
bits_for_seg = (
 1st  bit  # top left
 2nd  bit  # top right
 3rd  bit  # upper right
 4th  bit  # lower right
 5th  bit  # bottom right
 6th  bit  # bottom left
 7th  bit  # lower left
 8th  bit  # upper left
 9th  bit  # upper left slash
10th  bit  # upper middle
11th  bit  # upper right slash
12th  bit  # middle right
13th  bit  # lower right slash
14th  bit  # upper middle
15th  bit  # lower left slash
16th  bit  # middle left
)

seg_for_c[27] = {
    0b1111001100010001, // A
    0b0000011100000101, // b
    0b1100111100000000, // C
    0b0000011001000101, // d
    0b1000011100000001, // E
    0b1000001100000001, // F
    0b1001111100010000, // G
    0b0011001100010001, // H
    0b1100110001000100, // I
    0b1100010001000100, // J
    0b0000000001101100, // K
    0b0000111100000000, // L
    0b0011001110100000, // M
    0b0011001110001000, // N
    0b1111111100000000, // O
    0b1000001101000001, // P
    0b0111000001010000, //q
    0b1110001100011001, //R
    0b1101110100010001, //S
    0b1100000001000100, //T
    0b0011111100000000, //U
    0b0000001100100010, //V
    0b0011001100001010, //W
    0b0000000010101010, //X
    0b0000000010100100, //Y
    0b1100110000100010, //Z
    0b0000000000000000
};
```



- **writer (自行撰寫)**
  - 每隔一秒，就將新的字母寫到 driver 當中。
- **reader (助教提供)**
  - 每隔一秒，就去讀取 driver，透過 socket 將資料傳遞給 seg.py (http://seg.py) 程式。
- **seg.py (http://seg.py) (助教提供)**
  - 當作16段顯示器，將收到的資料，透過 Tkinter library 給顯示出來。

III. Illustration

- driver

```
insmod mydev.ko
```

- writer
  - <sec> 表示從該秒數開始倒數計時。

```
./writer <name>
```

- reader
  - <ip>, <port> 填入 VM 之 ip address。
  - <dev> 填入 mknod 建立的 character device 名稱，如 `/dev/mydev`

```
./reader <ip> <port> <dev>
```

- seg.py (http://seg.py)
  - <port> 填入 socket 聆聽的端口。

```
python seg.py <port>
```

IV. Note

- driver 的撰寫請參考 lab pdf 的範例程式碼。
- VM 上須安裝 Tkinter library，以利程式正確執行。

```
sudo apt install python-tk
```

- 由於 driver 與一般程式 (reader, writer) 所使用的定址空間不同，在傳遞資料的時候需要透過 **copy_from_user()**, **copy_to_user()** 這兩個 function 來協助完成。

V. Demo & Submission

- 助教會提供 demo.sh (http://demo.sh) 來協助同學 Demo。
- 執行.sh前務必先 “mknod”，並修改.sh裡的IP等資訊
- 編譯時須加上option “ --static ”，才能在ZC702正確執行
- 請將程式碼以下列的格式擺放與命名，以方便助教評分。

```
<學號>_eos_lab4
|-- Makefile
|-- mydev.c
|-- writer.c
|-- reader.c
|-- seg.py
|-- demo.sh
```

- 請將上述之資料夾壓縮為單一 zip 檔案，並上傳到 E3 上。

Lab 4 Hints

I. Preliminaries

- Review Lab 7 Linux Driver in laball

II. Environment Variable

- Export the cross compiler for ZC702 and activate it when cross compiling

```
export CROSS_COMPILE=arm-linux-gnueabihf-
source /tools/Xilinx/SDK/2018.3/settings64.sh
```

III. Network

- Network Environment
  - Check ip is correct as setting up at boot stage

IV. Network IP

- Enable network interface and set it’s ip address
- eth0 change if NIC name is different
- Ping if network is connected

```
ip link set eth0 up
ip addr add 192.168.0.202/24 dev eth0
ping -c 4 192.168.0.200
```

V. NFS

- Set up for NFS

```
mkdir /mnt
mount -o tcp,nolock 192.168.0.200:/home/lab616/nfs /mnt
# nolock — Disables file locking.
cd /mnt
```

- Permission denied

```
/home/lab616/nfs    *(rw,no_root_squash,no_subtree_check)
```

VI. Kernel Module

- How to insert or remove a kernel module

```
insmod mydev.ko
lsmod | grep mydev
# mydev 12288 0 - Live 0xbf000000 (O)

rmmod -f mydev
lsmod | grep mydev
# nothing here
```

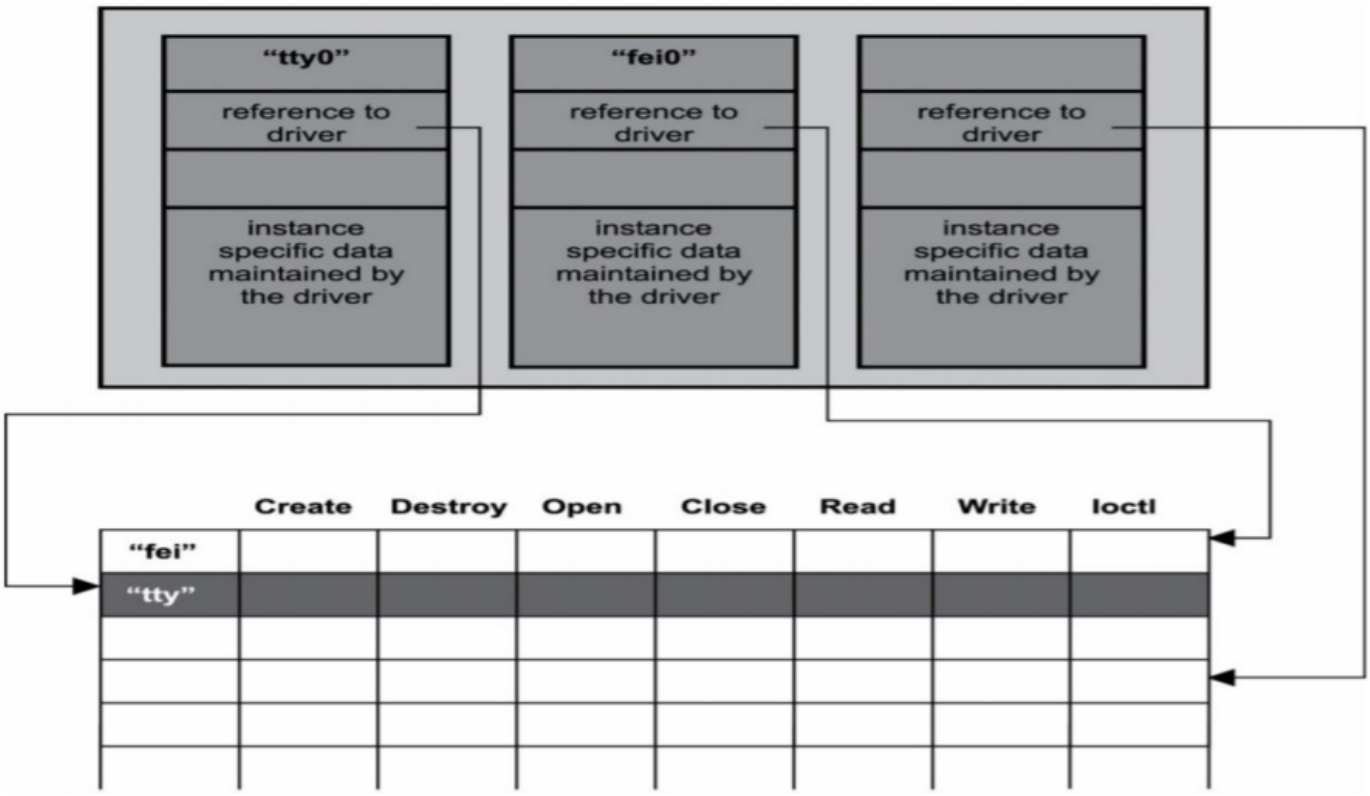
- Module dependency

```
mkdir -p /lib/modules/$(uname -r)
cp mydev.ko /lib/modules/4.14.0-xilinx/

modprobe mydev
modprobe -r mydev
```

VII. Create Character Device

```
mknod /dev/mydev c 244 0
```



VIII. Memory Access

- Copy from user/ to user is a kernel instruction

```
#include <linux/uaccess.h>
```

- copy_from_user (https://www.fsl.cs.sunysb.edu/kernel-api/re257.html) - Copy a block of data from user space.
- copy_to_user (https://www.fsl.cs.sunysb.edu/kernel-api/re256.html) - Copy a block of data into user space.