Lab 6

Lab6.1kernel編譯教學

本實驗將會教你如何編譯kernel並將kernel放入image中

/*助教所使用的環境為ubuntu 14.04 lts 32bit*/

1. 首先，大家開啟在實驗4中所做好的虛擬機器

2. 然後打開terminal

3. 安裝git 這一個工具

   sudo apt-get install git

4. cd到一個你想要的空資料夾（可以自己mkdir一個新的）

   我們所使用的軟體為linux-sunxi，略懂github的可以大略瀏覽一下以下網頁

   https://github.com/linux-sunxi/linux-sunxi

   執行以下指令

   git clone https://github.com/linux-sunxi/linux-sunxi.git

   過程可能有點久，大家可以稍微等待一下

5. 進入資料夾

   cd linux-sunxi

6. 修改Makefile

   export KBUILD_BUILDHOST := $(SUBARCH)

   ARCH ?= $(SUBARCH)

   CROSS_COMPILE ?= $((CONFIG_CROSS_COMPILE:"%"=%))

   改為

   export KBUILD_BUILDHOST := $(SUBARCH)

   ARCH ?= arm

   CROSS_COMPILE ?= arm-linux-gnueabihf-

7. 先把cubieboard A20的設定檔案載入

   make sun7i_defconfig

   PS:若發生make指令找不到，可以使用下列指令解決

   sudo apt-get install make

8. 把目錄底下編譯時產生的檔案清除(理論上應該是可以不用執行)

   make clear

9. 可能需要安裝ncurses5

   sudo apt-get install libncurses5-dev

10. 然後執行

    make menuconfig

在這裡出現了選單，我們可以在這裡調整我們kernel需要的東西
比較重要的部分為記得檢查

Device Drivers --->
Graphics support --->
{*} Support for frame buffer devices --->
<*> DISP Driver Support(sunxi)
-*- Reserve memory block for sunxi/fb
[*] Enable FB/UMP Integration
<*> LCD Driver Support(sunxi)
<*> HDMI Driver Support(sunxi)

是否有選取到,不然無法使用lcd與hdmi
大家可以自己遊玩一下！
設定完成直接exit就好了

11. 之後開始build

make uImage

等待一個很長的時間
PS.1:
如果你的處理器是多核心(virtualbox中的設定or實體機)
可以使用下列指令來加快速度

make -j4 uImage //-jXXX XXX為你的核心數，在此範例為4核心

PS.2:
若遇到”mkimage” command not found
這一個command是在u-boot-tools裡面
使用下列指令安裝

sudo apt-get install u-boot-tools

然後重新執行指令即可

12. 完成後我們便可以在arch/arm/boot/中找到uImage

ls -l arch/arm/boot/

可以看到uImage也就是我們的kernel編譯完成
如果沒有的話，可以看看顯示的錯誤訊息為何！

13. 然後按照Lab4的教學開啟了共享資料夾把img檔放入資料中並依照指令掛載

sudo mount -t vboxsf -o uid=$UID,gid=$(id -g) vm
~/anywhere-you-wants-folders

14. 然後執行

sfdisk -uS -l the-imgane-file-you-use.img

15. 而我們這次要掛載的是第一個磁區

mount -o loop,offset=$((sector大小＊2048))
the-imgane-file-you-use.img ~/anywhere-you-wants-folders

16. 再剛剛掛載的磁區中,我們可以看到有三個檔案,一個是script.bin跟uImage和uEnv.txt,這邊我們必須要跟剛剛編出來的uImage做替換,所以

    sudo rm ~folder-you-mount-img/uImage

    sudo cp ~folder-you-build-kernel/arch/arm/boot/uImage ~folder-you-mount-img/uImage

17. 把掛載的磁區卸除之後我們就完成了kernel的更換

Lab6.2如何開啟vga設定

本實驗會教大家如何在沒有vga接頭的情況下使用gpio來幫cubieboard新增一組vga接頭

1. 在上一個的實驗中，我們所掛載的第一個磁區中，其中一個檔是kernel，而另外一個檔為script.bin，這一個檔案為控制gpio設定的設定檔，但是他現在我們沒辦法解讀，我們需要另一個工具來解義他！

   git clone git://github.com/linux-sunxi/sunxi-tools.git

2. 進入資料夾

   cd sunxi-tools

3. 編譯fex2bin和bin2fex

   make fex2bin

   make bin2fex

4. 有了這2個程式我們就可以將.bin檔轉回可以看的fex檔

   ./bin2fex the-folder-you-mount-image/script.bin name-you-wanted.fex

5. 我們就可以看name-you-wanted.fex裡面有關於gpio的一些設定值

   vim name-you-wanted.fex

6. 在.fex檔中改變為下列值(理論上只有紅字要改)

   [disp_init]
   disp_init_enable = 1
   disp_mode = 0
   screen0_output_type = 4
   screen0_output_mode = 4
   screen1_output_type = 0
   screen1_output_mode = 4
   fb0_width = 1024
   fb0_height = 768
   fb0_framebuffer_num = 2
   fb0_format = 10
   fb0_pixel_sequence = 0

fb0_scaler_mode_enable = 0
fb1_width = 1024
fb1_height = 768
fb1_framebuffer_num = 2
fb1_format = 10
fb1_pixel_sequence = 0
fb1_scaler_mode_enable = 0
lcd0_backlight = 197
lcd1_backlight = 197
lcd0_bright = 50
lcd0_contrast = 50
lcd0_saturation = 57
lcd0_hue = 50
lcd1_bright = 50
lcd1_contrast = 50
lcd1_saturation = 57
lcd1_hue = 50

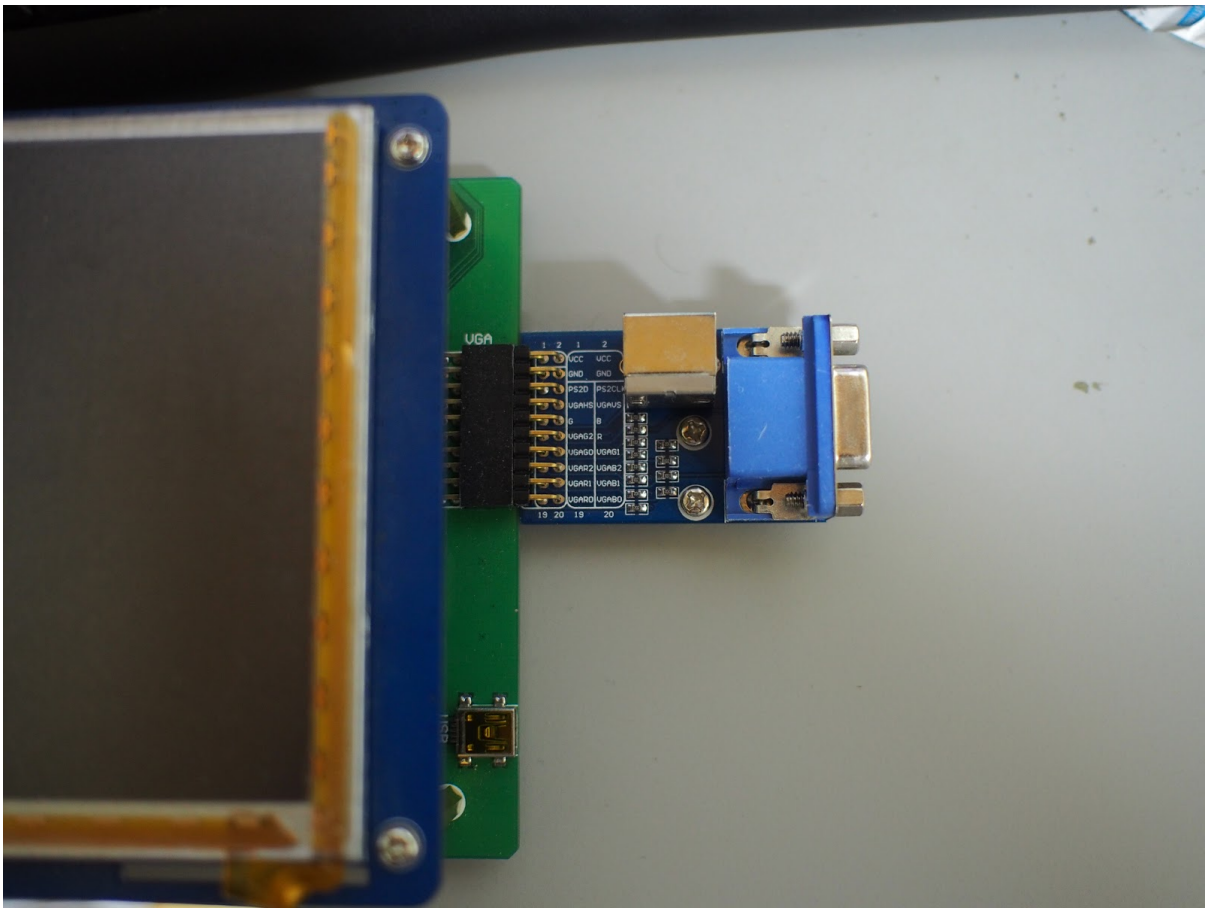
關於這些參數可以參考下列連結
http://linux-sunxi.org/Fex_Guide#disp_init_configuration

7. 編輯好後存檔,刪除script.bin檔然後建立新的檔按過去
   sudo rm the-folder-you-mount-image/script.bin
   sudo ./fex2bin name-you-wanted.fex
   the-folder-you-mount-image/script.bin

8. 卸載img檔
   sudo umount the-folder-you-mount-image/

9. 燒錄img檔至sd卡

10. 找到vga接頭後如圖接上插頭



11. 開啟電源便可以使用vga當作顯示螢幕了

Lab6.3io_control

本實驗我們將會簡單的教大家如何在cubieboard中加入一些io的control，例如蜂鳴器、按鈕led等等操作！

1. 首先取得waveshare_demo.7z檔，並解壓縮至虛擬機中，假設名子就叫做waveshare_demo，進入資料夾

   cd waveshare_demo

2. 然後找到io_control的driver

   cd driver/io_control_dev

3. 修改Makefile

   將下列這行

   KERNELDIR = XXXXXXXXXXX

   改為我們在Lab6.1中所編譯的kernel路徑

   KERNELDIR = XXXXXXXXXXX**/linux-sunxi**

4. 一樣我們先清除編譯過的文件

   make clean

5. 然後即可已開始編譯
   make
   ls看到io_control.ko為編譯成功
   PS若看到linux/init.h not found這一個錯誤可能是kernel的問題，雖然顯示錯誤，但是檔案都有出來，為正常現象

6. 掛載img檔,並將剛剛編譯好的io_control.ko放入img的磁區二(rootfs)中

7. 再一次進入waveshare_demo資料夾

8. 這次我們則是進入api資料夾
   cd API
   ls
   我們可以看到key_test和led_test和其他資料夾

9. 進入led_test
   cd led_test
   看到test_led.c

10. 編譯test_led.c
    arm-linux-gnueabihf-gcc test_led.c -o XXX-name-you-want

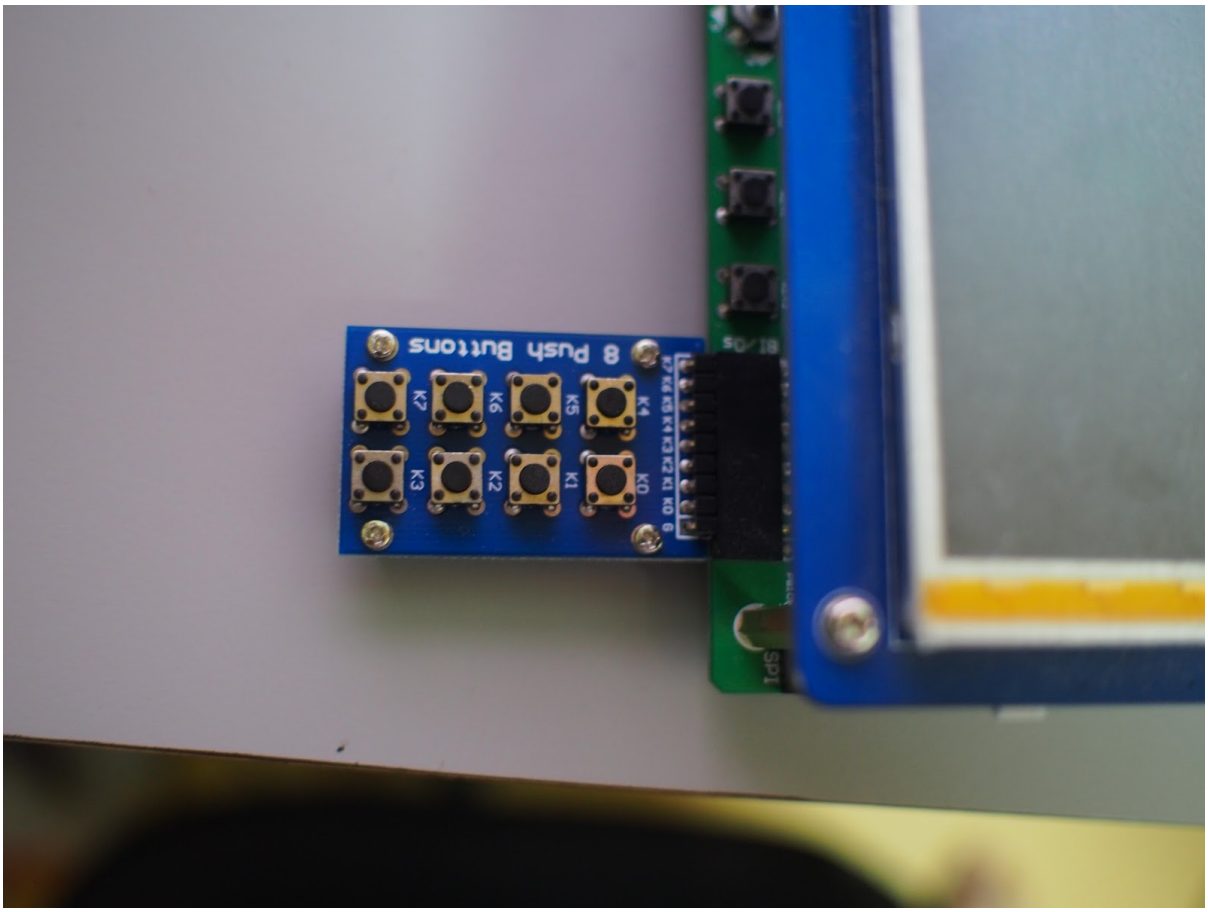
11. key_test以此類推

12. 然後將編譯好的程式放入img檔的磁區二(rootfs)中
    PS: 當然也可以按照實驗5的做法在cubieboard上編譯！

13. 燒錄img檔

14. 登入cubieboard後,把io_control驅動安裝進去
    insmod io_control.ko
    lsmod便可以看到載入的mod
    PS:遇到insmod error的可能為kernel版本不合,請使用最新的kernel(lab6.1的)

15. 首先執行led的測試程式
    ./XXX-led
    可以看到Led依序亮
16. 然後找到8 push buttons的模組,如圖插上版子



PS:3.3V不接

17. 然後執行key的test
    我們可以看到當我們按下按鈕時他就會顯示出我們所

############################
程式講解:

在這裡我們可以看一下io_control.c(也就是driver),為合這樣可以控制led與button呢?
先看到下列這一個function
io_control_read(struct file * file, const char __user * buf, size_t size, loff_t * ppos)
這一個function是用來控制這一個driver當他read的時候要如何讀取gpio和跟實際上的在檔案系統的值做溝通，這邊比較值得注意的是我們push_button所使用的所使用的pin腳為PI04-PI10和PG09和PG11，然後他讓這些pin腳設為pull up，而push button的電路圖為當按下按鈕的時候會變為ground，所以在程式的關鍵行

```
readdat= *pidat0;
readdat >>= i;
readdat &= 0x1;
if(readdat==0)
        a10_driver->rval= i;
if((readdat!=0) && (a10_driver->rval==i))
        a10_driver->rval= 0;
```

他使用readdat==0來判斷按下的按鈕,而如果**if((readdat!=0) && (a10_driver->rval==i))**成立時代表我這一個按鈕已經沒按了,但是rval還是現在這一個button的值,所以我要取消他
而這一段

```
if(copy_to_user(buf, &a10_driver->rval, size))
{
        printk(KERN_ERR "fail  copy_to_user!\n");
        return -EINVAL;
}
```

是將user在ret = read(fd, &read_pin,4)對應到的
read_pin ----- **a10_driver->rval**
4 ------ **size**
而ret為最後io_control_read的function return接回
當什麼都沒按的時候read都會是為0
所以我我們可以看到test_key.c中
fd = open("/dev/io_control",O_RDWR);
這一行我們將driver打開(linux中任何事物都是檔案)
ret = read(fd, &read_pin,4);這一個讀值
然後根據剛剛在driver回傳的值去做處理!
但這裡會有一個問題就是沒有辦法同時兩個以上的按鈕同時按下,我們的bonus就是來修改這一個檔案來讓他可以支援2個以上的按鈕被按下!

而在led控制中,我們可以先看io_control.c中的io_control_write(struct file * file, const char __user * buf, size_t size, loff_t * ppos)這一個function,
copy_from_user(&a10_driver->rval, buf, size)
首先把user輸入的值先輸入進去rval中
然後再一個switch中去控制我所要對應的功能
Led是在case 7 ~ 14,Case 15是讓led全關,而led所接的pin腳為PE04~11同理上跟io_control_read一樣,只是不一樣的是他這次是直接使用pedat0來控制腳位的0與1,而看到這裡我們可以看一下test_led.c檔,很明顯就與剛剛非常類似,我只要開檔,然後寫入case對應的值,他就會幫我們做case相對應的值所做的事情,大致上程式流程大概是這樣!

Demo:

Basic 1(30%):
將程式做成使用8個push button來控制led,例如按某一個按鍵就只亮某一個燈,按另一按鍵又亮另一燈。

Basic 2(30%):
使用網頁來控制led。

Bonus(20%):
將driver改為可以抓取2個已上button同時按的button&寫一個程式顯示什麼按鈕被按下了!

Bonus2(10%):
新增driver一個讓led全亮的mode， 並寫一個程式利用2個button來控制led全亮與全暗

參考資料
http://linux-sunxi.org/Linux_Kernel#Compilation
http://linux-sunxi.org/Fex_Guide#disp_init_configuration
補充:uEnv.txt他是boot的設定檔,可以參考以下連結
http://linux-sunxi.org/UEnv.txt
若想知道LCD是如何安裝的,可以參考這一篇
https://groups.google.com/forum/#!topic/cubieboard/qej8W6C945o