編譯 linux 0.11，並且使用 QEMU + GDB 調試 kernel (Ubuntu 11.04， GCC 4.5.2)

本篇為本人編譯 linux 0.11 心得，寫下來備忘

取得 source code

取得 source code 有很多方式，如果是直接下載原版的 code 的話，會很麻煩 (要使用舊版的 GCC 與 OS)， 網路上有改過的 code，適用於比較新的 GCC(linux-0.11-20110805.tar.gz)， 這邊有個適用 GCC 4.3.2 的，不過我是用更新的，適用 GCC 4.5.2 的 `linux-0.11-20110805.tar.gz`， 不過都是大同小異，如果有閒的話，可以把兩包抓來 diff 看看差在哪， 也可以去 oldlinux 的論壇裡面看

編譯

我這次的編譯環境為：`Ubuntu 11.04( linux kernel 3.0.2)`，`GCC 4.5.2`，而 source 我選擇`linux-0.11-20110805.tar.gz`，解壓縮後，使用 terminal，進入到該目錄，只要在有 MakeFile 這支檔案的目錄下，打入 make 就可以進行 compile， 如果編譯好的話，會在該目錄下，看到一個 Image 的檔案，就代表成功了

編譯問題排解

如果出現

```
make: *** [kernel/blk_drv/blk_drv.a] Error 2
```

這是一個明顯的宏語法錯誤，把 blk.h 的第 87 行的 #elif 改為 #else 就行了

如果出現

```
make: *** [tools/system] ?? 1
```

就把 GCC 加入編譯選項

```
-fno-stack-protector
```

(只要在 make.header 中的 CC 選項加入，就可以全部加入 ， 例如

```
CC = gcc -fno-stack-protector)
```

如果再打入 make 不行的話，可以先下 make clean 指令後，再下 make

如果想看每一個檔案所編譯出來的 Object file (附檔名為.o)的 assembly 的話，可以使用 objdump 來觀察 如：

```
objdump -S test.secton.o (-S 的參數式大寫，這邊大小寫有差別)
```

也有圖形化介面的 objdump， 叫 dissy 也是很好用的

如果想看 Hex file的話，可以使用 `hexdump'，如

```
hexdump -C boot.img
```

-C 參數可以以 BYTE 的方式顯示

模擬測試

使用 x86 emu 把 linux 0.11 裝起來，網路上趙博士是使用 bochs，不是很會用

我選擇使用 qemu，鍵入

```
qemu -m 16M -boot a -fda Image -hda ./../rootfs/hdc-0.11-new.img
```

而`-fda Image`：代表你把 Image 執行目錄下

而`-hda ./../rootfs/hdc-0.11-new.img`：代表你把 HD img，放在相對目錄”上一層”的`rootfs/`下

而`hdc-0.11-new.img`：為一模擬 hd 的檔案，可以在趙博士所提供的`linux-0.11-devel-060625.zip`找到

而`../hdc-0.11-new.img`：視你把該`hdc-0.11-new.img`檔放在哪而定

其實把 makefile .header 打開，裡面其實有 debug，也就是如果要 run qemu 的話，只要打 make debug 就會執行..

QEMU + GDB

鍵入以下

```
qemu -m 16 -boot a -fda Image -hda ./../rootfs/hdc-0.11-new.img -s -S -gdb tcp::1234
```

這邊有兩個 `-s` 與 `-S`，分別代表不同意義

`-s`(小寫 s)：運行虛擬機時將 1234 端口開啟成調試端口，供 eclipse 網絡調試時使用

`-S`(大寫 S)：啟動虛擬機時要「凍住」虛擬機，等待調試器發出繼續運行的命令

`-fda`：在 qemu 中，是使用檔案來模擬磁碟的，這邊的意思是，使用 Imgae 這個檔案來當作 floppy A

`-m`：設定模擬的記憶體有多大，這邊設定記憶體大小為 16MB

順利的話，應該會顯示黑畫面，代表目前 QEMU 被凍住，等待 GDB client

在開另一個 terminal，鍵入

```
gdb tools/system
```

即可進入到 gdb client

先載入除錯的 symbol

```
(gdb) file tools/system
```

連線至遠端

```
(gdb) target remote localhost:1234
```

下中斷，停在 `CS = 0x7C00` 的地方，也就是 BIOS 把 MBR 載入的地方

```
(gdb)  br *0x7c00
```

在此時，bios 把控制權正式的交給了 linux，也就是說這裡開始就是我們自己控制的地方

而 0x7C00 對應的 code 應該是 bootsect.S

觀察0x7DFE 與 0x7DFF的值是否為0x55，0xAA

```
(gdb) x/16b 0x7DF0
```

因為這邊是組合語言，所以要用 si 來單步

```
(gdb) si
```

下中斷

```
(gdb) b main
```

執行至中斷 ，則就會執行到 main 中

```
(gdb) c
```

列出目前的 code(list)

```
(gdb)l
```

介紹一下 gdb 指令

`b: 下中斷點`

`continue(c) 繼續執行直到下一個中斷點或結束`

`list(l): 列出目前上下文`

`step(s): 單步 (會進入 funciton)`

`next(n) : 單步 (不會進入 funciton)`

`until(u) 跳離一個 while for 迴圈`

`print(p): 顯示某變數，如 p str`

`info b :u 列出目前中斷點`

`info register : 顯示 CPU 的 register`

QEMU 問題排解

而 qemu 要順利打開可能會需要用到 kvm

可以使用 lsmod|grep kvm 看看 kvm 模組有沒有被載入

如果沒有的話，可以打

```
sudo kvm-ok
```

試著載入看看

如果再不行，看看 cpu 是否支援 VT or 機版 bios 是不是沒打開

`intel-vt ( amd cpu 是 amd-v)`的選項

重開機打開後，應該就可以使用 qemu 了

參考資料

可供GDB源碼調試的用GCC 4.X編譯的Linux 0.11實驗環境

目前已經成功地把linux-0.11移植到gcc 4.3.2

《Linux內核完全註釋》閱讀筆記——搭建實驗環境(轉載) Welcome to OldLinux

Eclipse CDT + QEMU 調試linux內核

GNU偵錯器