Computer Organization and Assembly Languages

Final Project: Real Mode Operating System

B92203051 化學四 林偉楷

摘要

設計一個真實模式的作業系統，以瞭解計算機硬體的運作方式。

研究動機

經過一學期計算機組織與組合語言課程，從 TOY Machine[1] 的硬體架構和機械語言開始，逐漸瞭解到電腦是如何架構起來的。但 TOY Machine 只是一個模型，和目前實際使用的電腦差異很大；而學過一些 Intel 組合語言的概念以後，更對兩者間的差異——如基礎 I/O 運作——感到好奇。因此希望透過實際寫一個作業系統的方式，瞭解這些細節。

研究方法

用 Intel 組合語言寫一個真實模式（real mode）的作業系統，由軟碟開機。軟碟上有一個簡單的檔案系統， 可透過作業系統處理檔案。作業系統提供簡單的命令列做為使用者界面， 並內建數個操作檔案的指令。作業系統也能執行檔案系統中的其他檔案， 執行結束後回到命令列界面。

實作時使用的工具程式為 GNU 的 gcc preprocessor、 as assembler、 ld linker[2]，使用 AT&T 組合語言語法。 測試時則使用 QEMU[3] 模擬器。部份程式碼參考 GRUB[4]。

研究結果

在電腦開機的時候， BIOS 會先從開機磁碟讀取 512 bytes，即 Master Boot Record (MBR)[5]。 BIOS 將 MBR 載入記憶體，並且跳過去執行。 MBR 的內容除了開機程式碼（boot loader）外， 通常還包括磁碟分割表，以及最後兩個 byte 的內容為 0x55 和 0xAA，做為 BIOS 辨識用。另外 BIOS 固定將 MBR 放在記憶體位址 0x7C00[6]，因此 linking 時需特別指定。

為簡化實作上的困難，檔案系統將 1.44 軟碟規劃成 160 個區塊， 每一個區塊為軟碟上連續的 18 個 sector，且每個區塊只能放一個檔案； 所以單一個檔案大小上限為 9 KB。 每一個檔案的檔名和大小儲存在一個檔案表中，規定檔名最長為 6 bytes。 第一個區塊存放 MBR 和檔案表，第二個區塊則是作業系統的核心。

記憶體的規劃上， real mode 使用 segment register 和 offset 相加的方式定址，總共可定址的空間為 1MB。 作業系統本身使用的空間為 0x10000 以前： 0x5000 開始放檔案表，0x7C00 放 boot loader， 0x8000 放核心；使用者執行的程式從 0x10000 開始， stack 則從 0xBFFFF 開始填。

規定好基本的磁碟和記憶體配置以後，實作上就很容易了。 因為核心在磁碟裡的位置

是固定的， boot loader 只需要把磁碟內容讀取到記憶體中然後跳過去。核心開始執行時，顯示提示字元並且讀取指令。取得輸入指令後先檢查是否為內部指令， 如果不是就尋找磁碟中有沒有同名的檔案，找到就載入到記憶體 0x10000 處並跳過去執行。被執行的程式可以使用所有的系統資源， 若要結束就跳回 0x8000 即可；然後核心再繼續執行下一個輸入的指令。 內部指令也是執行完就回到命令列，共有 5 個：

1. `ls`: 顯示磁碟內所有檔案
2. `nf`: 建立一個空的新檔案
3. `mv`: 將檔案改名
4. `cp`: 複製檔案
5. `rm`: 刪除檔案

其中 `cp` 是內部指令中唯一會修改實際檔案內容的， 其他都只是在檔案表上查詢和標記而已。

所有 I/O 的動作都是透過 BIOS 的 interrupt 完成的：使用 INT 0x13 存取磁碟內容， INT 0x10 把訊息輸出到顯示器，還有 INT 0x16 取得鍵盤的輸入。

因為可執行檔是直接從磁碟中複製到 0x10000 處執行， 這些程式連結的時候也必需指定記憶體位址。規定可執行檔執行時， code segment (CS) register 為 0x1000，所以連結的時候指定從 0x0000 的位置開始才能正確被執行。

組譯和測試

組譯並建立磁碟映像檔：

```
./build
```

將磁碟映像檔寫入軟碟片：

```
dd if=image of=/dev/fd0
```

然後即可用軟碟開機測試，或者直接用 QEMU 讀取磁碟映像檔進行模擬。 除內部指令外，唯一的可執行檔是 `hlwd`，印出 "Hello world!" 然後結束。軟碟開機測試時，部份 PC 無法執行， 推測是 BIOS 規格差異造成的。

結論

作業系統設計的時候，規格的制定非常重要，對系統效能影響很大； 也就是說，定規格不見得比實作簡單，需要考慮各種情況。 像是這個作業系統只支援軟碟機，如果要支援硬碟機， 檔案系統就一定要大幅修改。

最後， PC 上的 BIOS 可以處理許多複雜的 I/O 工作，相較之下 TOY Machine 必需透過 memory map 或者 instruction 處理 I/O，是非常大的差異。

參考資料

1 http://www.cs.princeton.edu/introcs/xtoy/

2 gcc: http://gcc.gnu.org/onlinedocs/; as: http://sourceware.org/binutils/docs-2.17/as/index.html; ld: http://sourceware.org/binutils/docs-2.17/ld/index.html
3 QEMU: http://fabrice.bellard.free.fr/qemu/
4 GRUB: http://www.gnu.org/software/grub/
5 MBR: http://en.wikipedia.org/wiki/Master_boot_record
6 VnutZ, *BootStrap Tutorial*, http://perso.orange.fr/pierrelib/bootstrap/VnutZ_Bootstrap_Tutorial.html