Linux 下網路驅動程式追蹤

Ⅰ. 實驗目的

瞭解並實地追蹤 Linux 下網路卡驅動程式（adapter driver）、協定驅動程式（protocol driver）的原始碼，內容包括：（1）追蹤開機程序中網路 Process 的始末，（2）研究 Linux 中的軟體架構並進行核心編譯，（3）追蹤核心（kernel）中的原始碼，包括網路卡驅動程式與協定驅動程式。

實驗報告的內容應包含：實驗題目、參與人員及單位、目的、設備、方法、記錄、問題討論、心得。報告各部分內容應是經過討論、整理、濃縮後，重新詮釋而寫出，切勿全部剪下、照單全貼。

所列實驗方法爲參考用，若有更好的方法請在報告問題討論裡的「自問自答題目」裡詳述步驟，會有額外的加分。

Ⅱ. 實驗設備

本實驗需要的硬體請自備。以下所列項目爲最低需求，請儘可能使用較新的軟硬體。

一、硬體

項目	數量	備註
個人電腦 PC	1	Intel x86-compatible processors 及 256MB 記憶體 (或以上之配備)
網路卡	1	目前市面上流行的網路卡皆能適用
軟式磁碟片	1	1.44MB floppy

二、軟體

項目		數量	備註
Linux	Fedora Core 5	1	本實驗採用 Redhat 系統的 Linux 來說明。採用

			其他 distribution 或其他公開原始碼且與 UNIX 相容的版本（如 FreeBSD）亦可。Linux 可由各大 FTP 站取得[1][2]。

Ⅲ. 背景資料

一、Linux 的背景

Linux 是一套完全免費、32 位元多人多工的 UNIX 相容系統。它最早是在使用 Intel x86 CPU 架構下的 PC 發展，如今它已能於多種 CPU 架構下執行，如 Compaq Alpha AXP、Sun SPARC、Motorola 68000 系列、MIPS、Power PC、ARM、SuperH 等。它不但同時相容於 System V 和 BSD UNIX，更符合 POSIX 標準。最重要的是，它的原始碼完全公開，更新非常快速，所以非常適合學術界研究使用，且將是許多 embedded system 採用之作業系統。

Linux 的起源最早是在 1991 年 10 月 5 日由一位芬蘭的大學生 Linus Torvalds 撰寫了 Linux 的核心程式 0.0.2 開始的，然而其後續發展幾乎是拜網際網路上世界各方高手貢獻而成。由於 Linux 具有免費、效能高、原始碼公開等特性，目前流行的程度非常迅速，除了許多軟硬體公司已開始移植軟體到 Linux 上，更有許多相關的發展計畫，如 GNOME、KDE、CLE 等等正在進行著。在商業界裡更有許多公司行號早已以 Linux 當作 Server。Linux 作業系統可以說是軟體的一股潮流、Bill Gates 揮之不去的夢魘。

二、Linux 的核心功能[3]

Linux kernel 的功能與其他 UNIX 系統的 kernel 類似，主要是處理「Process Management」、「Memory Management」、「File System」、「Device Control」、「Networking」等事項：

1. Process Management

   process 的起始（create）、終結（destroy）與 process 間的通訊（Inter-Process Communication）都由 kernel 來管理。

2. Memory Management

   kernel 建立一個 virtual addressing space 的記憶體給所有的 Process 使用。應用程式只需利用 malloc、free 等的 function call 即可使用記憶體資源而不與其他 process 衝突。

3. File System

   Linux 非常仰賴 File System 的觀念，幾乎所有東西（如虛擬終端機、周邊裝置等）皆可以被對應到檔案中。

4. Device Control

Linux 在 device driver 方面與 DOS 很不一樣的是，把 device driver 併在 kernel 裡。所有硬體的驅動程式都跑在 kernel space，所以欲加入新的硬體，大半都必須重新編譯核心。有一辦法是將驅動程式寫成 module，就可以動態地載入、移除驅動程式而不需重新編譯。

5. Networking

由於網路的功能不能只針對某些 process，使得網路的部份需由作業系統直接負責。一個封包從 interface 進來後，中間尚須經過層層的 protocol stack(如 TCP、IP 等)才分派到 user space 的應用程式；這裏 Networking 專指中間 protocol stack 的部分。

三、Linux 的網路架構

除了 GNU（GNU's Not Unix）中眾多的軟體可以使用外，Linux 還可以當作 router、firewall 等網路設備。由於其網路功能非常強大，又公開原始碼，使我們對網路的運作流程產生無比的興趣，本實驗的目的就是追蹤 Linux「網路卡驅動程式」、「網路協定驅動程式」、「網路應用程式」之間的互動關係。

由於「驅動網路卡」的工作屬於核心功能中「device control」的部分，而「協定堆疊」屬於核心功能中「Networking」的部分，我們得知此兩者在 Linux/UNIX 的環境下，是存在於核心程式（kernel）中。

四、檢視核心原始碼

在此我們僅檢視有關網路方面的核心原始碼，也就是「網路卡驅動程式」與「協定驅動程式」的原始碼：

1. 網路卡驅動程式
   A. **Hardware Dependent 的部分：**在 kernel source tree 根目錄下的 drivers/net/裡有各種已被支援的網路卡驅動程式原始碼。如果是 NE2000 Compatible 的 Chip，原始碼爲 ne.c；若是 Intel EtherExpress PRO 10/100Mbps 的網路卡，原始碼爲 eepro100.c，以此類推，詳細的網路卡支援都可在 source tree 下之原始碼中找到。
   B. **Hardware Independent 的部分：**MAC 層的運作可在 source 根目錄的 net/ethernet/中找到。
2. 協定驅動程式

   核心原始碼在 source 根目錄的 net/ipv4/，例如 ip_output.c 中有許多重要的 function 如 ip_build_and_send_pkt，ip_queue_xmit 等[1]。

五、開機程序始末[5]

一個典型的 Linux 開機程序有數個步驟：

1 **核心載入**

在 Linux 中通常以 grub 來載入核心.，/etc/grub.conf 爲其設定檔。

2 **硬體偵測與設定**

核心載入後執行硬體偵測設定、初始化核心的工作。網路卡及協定驅動程式皆在此載入。

3 **產生 System Process（init 等 process）**

硬體偵測完後產生 init 這個 user space 的 process。init 是所有其他 process 的老祖宗，其初始化是經由讀取/etc/inittab 設定檔來開始執行系統初始化的 script。

4 **執行系統初始化的 script**

/etc/inittab 是 init 的設定檔，裡頭有參考到許多其他的/etc/rc.d/中的 script，其中第一會先參考到/etc/rc.d/rc.sysinit，再來會參考到的是/etc/rc.d/rcX.d，X 爲 0~6，每一種代表一種開機模式。Internet 的 daemon（xinetd）就是在這些 script 中載入的[2]。

5 **進入 multi-user 的 session**

由 init 衍生出 getty，再衍生出一個 login process 來讓使用者登入。

IV. 實驗方法

安裝 Linux

安裝 Linux 非常簡單，用光碟直接開機安裝，或用 FTP 等就可以灌好。坊間有許多參考書籍介紹安裝這部分，網路上也有很多 site 有說明文件，在此就不贅述。

啓動 Linux 的網路功能

在 Fedora 5 [8]，只需要在安裝的時候，設定好 IP address、netmask、getway、DNS server（Primary nameserver）等等，即可在後開機直接啓動網路功能，若想要在開機後重新設定網路功能，則可利用 ifconfig 及 netconfig 指令，或是使用 x window 中的 System->Adminstration->Network 來設定，設定完不需重新開機，只需要下“service network restart”指令即可將網路功能重新啓動。若是網路卡十分特殊，則需要去原廠網頁抓此網路卡的驅動程式並安裝，最後再照上面步驟執行即可。

觀察開機程序

觀察開機的方法有很多，主要可以列成幾點：

1. **觀察開機螢幕：**開機時可馬上記錄螢幕顯示，先用 Scroll Lock 將螢幕畫面暫停，再利用 shift+PageUp 上下觀察
2. **利用 dmesg 指令:**開機結束進入帳號後輸入 dmesg 指令以觀看開機訊息。
3. **觀看 log 檔：**透過 syslogd 記錄到 log 檔，詳細情形見下文。

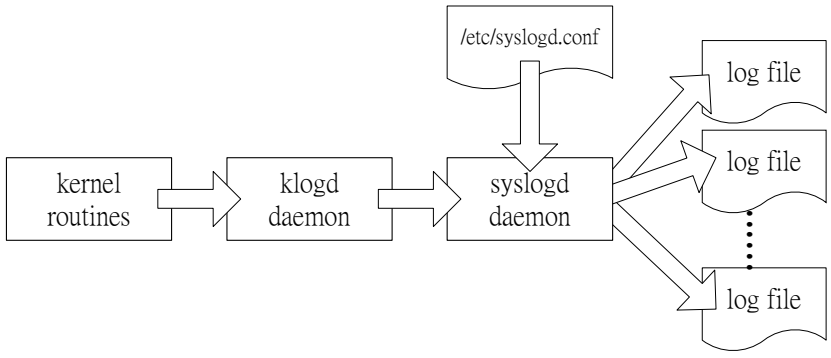
追蹤核心中網路驅動程式之方法

在本實驗裡，我們將執行 ping 來觸發 kernel 中的網路處理模組，並利用下列提供的兩個方法擷取執行的過程。

追蹤核心程式與追蹤普通 user space 的程式有很大的不同，在此提供兩個方法：利用「kernel 之 printk 透過 syslogd 來記錄執行程序」、利用「gdb 之 remote debugging 致使 kernel 可以 step by step 執行」，而實驗步驟則專注在第一種方法。

1. **利用 printk 產生訊息：**在 kernel space 中無法像一般在 user space 寫程式一樣，利用 printf 將訊息列印到標準輸出，因爲 printf 是一個讓 user space 程式使用的 function。Kernel 中要利用 printk 來將訊息列印出來，其與 printf 不同的是，他沒有處理浮點數的能力，另外就是每一個 printk 還伴隨著一個 log level，可以說就是此 message 的 priority。Linux 系統有一 klogd daemon，可以記錄 kernel 的 message。如果系統上尚有 syslogd，就可以很方便地攔截 klogd 的 message，並根據不同的 log level 記錄在不同的檔案或顯示在不同的 console（見圖 2-1）。

   syslogd 的設定檔是/etc/syslog.conf，把 kern.*的加上註解，並加上一行 ”kern.=info /var/log/kern_info” 就可以將所有 log level 爲 KERN_INFO 的 message 寫在/var/log/ker_info 的檔案。



【圖2-1】klogd與syslogd合作記錄kernel產生的message

使用 printk 的方法很簡單，其格式爲 printk(loglevel “formatted string”)[3]。log level 的值詳見表 2-1，可由 kernel.h 中找到其定義。

Log level	意義
KERN_EMERG	System is unusable
KERN_ALERT	Action must be taken immediately
KERN_CRIT	Critical conditions
KERN_ERR	Error conditions
KERN_WARNING	Warning conditions
KERN_NOTICE	Normal but significant condition
KERN_INFO	Informational
KERN_DEBUG	Debug-level messages

【表 2-1】Kernel message 的 log level

在 TCP/IP 的 protocol stack 中加入「適量」的 printk function call，即可得到一些有意義的結果。

2. **利用 debugger：**

在 trace 程式時我們通常會用 debugger 來設 breakpoint，並 step by step 地去執行程式，然而在 debug 對象是 kernel 時，可就享受不到這種便利了。畢竟 debugger 也是 user space 的 application，要 debug 到 kernel space，就必須要透過一些其他的手段。kdebug 是一個好用的工具，其利用 gdb 的 remote debugging interface 在「run-time」來做到一些 debug 的動作，像是「更改欲偵測對象之 data」、「呼叫函式（如呼叫 printk 印出某個變數目前的值）」等。

雖然 kdebug 可以在 run-time 做到上述的動作，卻仍不行做到設定中斷點或是單步執行，因爲一台機器總不能邊把自己的 kernel 停掉，卻要他繼續執行 debugger 來看一些變數值等類的事。要做到這樣的事可以透過 remote debugging 的方式，由一台電腦跑 gdb，另一台電腦跑 kernel，而兩台之間利用 RS-232 串連起來溝通。由於此種方法較複雜，詳情請參考[3][4]。

[3] 注意 printk 中的 loglevel 和後面的字串之間並無逗點。

V. 實驗步驟

1. 追蹤開機訊息
   1.1. 在開機以 root 登錄後輸入 dmesg > file，將開機訊息 dump 到檔案裡並記錄檔案內容【記錄 1】。
2. 編譯核心及設定 IP 位址
   2.1. **取得核心：** 請在 Linux 下到 FTP site [6]抓取核心原始碼至/usr/src
   2.2. **解開核心：**在您的 home directory 下使用 tar xvzf linux-2.x.x.tar.gz(檔名視 kernel 版本而定[4])解出 kernel source tree，並詳讀其根目錄下的 README 檔。
   2.3. **設定核心：** 在編譯前得先設定好 kernel 要有哪些功能。至 kernel source tree 根目錄下執行 make config（一問一答設定）或 make menuconfig（依選單選擇設定）進入設定 kernel 的階段。另外，如果你有安裝 X 視窗環境，利用 make xconfig 會有比較好的介面。在下面我們專注在網路方面的 kernel configuration：
      2.3.1. 在 Loadable Module Support 的大項裡：
         2.3.1.1. Enable loadable module support 選 YES 可以讓系統有能力動態載入模組，達到節省資源使用的目的。不過若光有選這個，要用到此 module 時尚須手動利用 insmod 或 modprobe 指令載入。
         2.3.1.2. Automatic Kernel module loading 選 YES 可讓 kernel 認爲要用到某 module 時自動將 module 載入，不需手動用 insmod 或 modprobe 載入。
      2.3.2. 在 Networking->Networking Option 大項裡：
         2.3.2.1. TCP/IP networking 選 YES 可讓 kernel 支援 Internet 的功能，也就是有了 TCP/IP 的 protocol stack。
      2.3.3. 在 Device Drivers->Network Device Support 大項裡：
         2.3.3.1. Network Device Support 選 YES 以支援網路卡。
         2.3.3.2. Ethernet (10 or 100Mbps)->Ethernet (10 or 100Mbps) 選 YES 可讓 kernel 支援 IEEE 802.3 制訂的標準（如 10BASE-T、100BASE-TX 等）。
         2.3.3.3. Ethernet (10 or 100Mbps) 下可讓你選擇您的網路卡型號

      如此設定完後，會產生適當的 makefile 供你編譯。

   2.4. **輸入 make clean：**將之前編譯過的檔案刪除，如果是第一次編譯則不需此步驟。

2.5. **輸入 make：**編譯新的 kernel image。若編譯不成功或是最後掛不起來，則可能要重新選擇 config 中的項目。若編譯成功 image 會放在 kernel source tree 下的 arch/i386/boot/下的 bzImage。

2.6. **安裝 module：**利用 make modules_install 將新的 module 拷貝到/lib/modules 下。

2.7. **移動新的 kernel image 到/boot 目錄：**將 kernel source tree 下 arch/i386/boot 中的 bzImage 搬到/boot 下面後並更名爲 vmlinuz-2.6.15.1（指令爲"mv /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-2.6.15.1"）

2.8. **移動新的 System map 到/boot 目錄：將 kernel source tree 下的 System.map 搬到/boot/System-2.6.15.1.map**

2.9. **切換到/boot 目錄下，製作 ramdisk images 給 preloading modules**：mkinitrd initrd-2.6.15.1.img 2.6.15.1

2.10. **編輯 Linux loader 的設定檔：**更改/etc/grub/menu.lst，新增如下幾行（此處只要仿照原本能夠開機的設定，再稍微更改即可；不同實驗者之設定可能不盡相同）：

```
title Mykernel (2.6.15.1)
        root (hd0,5)
        kernel /vmlinuz-2.6.15.1 ro root=LABEL=/ rhgb quiet
        initrd /initrd-2.6.15.1.img
```

2.11. **設定 Internet 參數：**在執行本步驟前雖然 kernel 已經支援 Internet 的各項模組，但是若不加以設定正確的參數，仍然無法連上 Internet。通常在安裝 Linux 時就會設定 IP address、Netmask 等，系統也會建好 routing table。不過當作一些比較複雜的事時（譬如插很多張網路卡，或這台 Linux 要當 router 等），就會需要一些 tool 來協助管理這些參數了。

2.11.1. **設定網路卡參數**：

假設欲設定的網路卡參數如表 2-2 所示：

參數	參數值
Interface	eth0
IP address	140.113.88.181
Broadcast address	140.113.88.255
Netmask	255.255.255.0

【表 2-2】設定網路卡的參數

你可以經由指令 "ifconfig eth0 140.113.88.181 netmask 255.255.255.0 broadcast 140.113.88.255" 來設定你的 eth0 interface，或是用如之前所述的 netconfig 或 x window 介面。

2.11.2. **設定 routing table**：

平常不需要使用到這個指令，但是如果插了好幾張網路卡，便需要這個指令來協助管理 routing table，以告知應用程式要從哪一個 interface 送出去。經由設定 routing table，指明哪些 IP 位址往那個 interface 送，就可以解決這樣的問題。如果要加一個 default gateway 的 entry 到 routing table，鍵入 “route add –net default gw 140.113.88.254 dev eth0”；如果要讓所有通往 140.113.23.*的封包都從 eth1 interface 出去，鍵入 “route add –net 140.113.23.0 dev eth1” 即可，或是直接使用 x window 下 System->Administration->Network->Devices tab，點擊 device，設定 Route。

3. 追蹤核心程式碼網路驅動程式

3.1. **設定 syslogd**：編輯/etc/syslog.conf，把 kern.*的那一行加上註解，並加上一行“kern.=info /var/log/kern_info”。存檔後重新開機，syslogd 就會可以將所有 log level 爲 KERN_INFO 的 message 寫在/var/log/ker_info 的檔案。

3.2. **追蹤 ping 送出 ICMP echo 的流程**：[7]

爲追蹤 ping 的工作流程，請到 kernel source tree 將下列檔案的函示中加上註解，亦可於欲追蹤之函示上加入註解

編號	檔名	函式名稱	備註
1	net/ipv4/icmp.c	xrlim_allow()	
2		icmpv4_xrlim_allow()	
3		icmp_reply()	
4		icmp_send()	
5		icmp_unreach()	加在 ICMP_NET_UNREACH ICMP_HOST_UNREACH ICMP_PORT_UNREACH ICMP_PROT_UNREACH
6		icmp_redirect()	
7		icmp_echo()	
8	net/ipv4/ip_input.c	ip_local_deliver()	
9		ip_rcv()	
10	net/ipv4/ip_output.c	ip_build_and_send_pkt()	
11		ip_output()	
12		ip_queue_xmit()	
14		ip_append_data()	
15		ip_append_page()	

16		ip_fregment()	
17		ip_reply_glue_bits()	
18		ip_send_reply()	
19	/net/ipv4/arp.c	arp_send()	
20	/net/ipv4/af_inet.c	inet_listen()	
21		inet_create()	
22		inet_release()	
23		inet_bind()	
24	/net/ipv4/route.c	ip_route_output_slow()	
25		ip_route_output_flow()	
26		ip_route_output_key()	
27	/net/ethernet/pe2.c	make_EII_client()	
28	/net/ethernet/eth.c	eth_header()	
29		eth_rebuild_header()	
30		eth_type_trans()	
31		eth_header_parse()	

之後到 kernel source tree 的根目錄下重新執行「編譯核心」的步驟 (2.6~2.9 及 2.11)，如此僅會重新 compile 有被更動過的檔、link 出新的 kernel 並更新 kernel，接著就可以實驗之後的步驟。當然，你也可以視需要自行添加 printk。

3.2.1. **選擇 ping 的待測機器：**爲嚴格定義待測機器位址，請按照下列公式推出在你環境下的近端、遠端機器之 IP 位址。

公式： (H and M)＝(R and M)＝(N1 and M)＝(N2 and M)≠(F and M)

公式中的 and 代表作 bitwise 的 AND 運算。H、M、R、N1、N2、F 的意義請參照表 2-3。將原已設定的 H、R、M 與選出的 F、N1、N2 記錄下來【記錄 2】。

	狀態	IP address（32-bit）
本機位址	已連上 Internet	H（host）
本機 Netmask	-	M（mask）
router 或 gateway 位址	已連上 Internet	R（router）
選擇的遠端（跨 subnet）待測機器位址	已連上 Internet	F（far）
選擇的近端（同 subnet）待測機器位址	已連上 Internet	N1（near）
選擇的近端（同 subnet）待測機器位址	尚未連上 Internet	N2（near）

【表 2-3】H、M、R、N1、N2、F 的意義

3.2.2. **清除本機 ARP table：**

重新開機以後，在文字模式下開啓三個 Virtual Console（Ctrl+**Alt+F1**、Ctrl+**Alt+F2**、Ctrl+**Alt+F3**）分別以 root 登入，第一個 Console 專門用來下指令，第二個 Console 專門用來看 log 檔，第三個專門用來看 ARP table，之後會用到的指令請參考表 2-4。

指令	Virtual Console	指令內容
A	1	ping –c 1 xxx.xxx.xxx.xxx（ping 待測機器一次）
B	2	more /var/log/kern_info（編輯 log 檔） cat /dev/null > /var/log/kern_info（清除 log 檔）
C	3	cat /proc/net/arp（檢視目前系統 ARP table） 或 指令 arp

【表 2-4】清除本機 ARP table 的指令

3.2.3. **ping 遠端機器**：

3.2.3.1. 下達指令 C 觀看 ARP table，記錄目前已有的 Entry【記錄 3】。

3.2.3.2. 下達指令 B 中的第二個指令清除 log 檔的內容。

3.2.3.3. 緊接在步驟 2 後，下達指令 A 測試遠端已可上 Internet 的機器（位址爲 F）。

3.2.3.4. 下達指令 B 觀看 log 檔，記錄後將其內容全部清掉【記錄 4】。

3.2.3.5. 下達指令 C 觀看 ARP table，記錄目前 table 中有的 Entry【記錄 5】。

3.2.4. **ping 近端機器**（ping 在同一 subnet 的機器，不透過 router）：

3.2.4.1. 下達指令 B 中的第二個指令清除 log 檔的內容。

3.2.4.2. 緊接前一步驟，下達指令 A 測試近端已可上 Internet 的機器（位址爲 N1）。

3.2.4.3. 下達指令 B 觀看 log 檔，記錄後將其內容全部清掉【記錄 6】。

3.2.4.4. 下達指令 C 觀看 ARP table，記錄目前 table 中有的 Entry【記錄 7】。

3.2.4.5. 重複步驟 3.2.4.2～3.2.4.5【記錄 8】。

3.2.5. **ping 近端不存在或未上網機器**：

3.2.5.1. 下達指令 B 中的第二個指令清除 log 檔的內容。

3.2.5.2. 緊接前一步驟，下達指令 A 測試近端不存在或未上網機器（位址爲 N2）。

3.2.5.3. 下達指令 B 觀看 log 檔，記錄後將其內容全部清掉【記錄 9】。

3.2.5.4. 下達指令 C 觀看 ARP table，記錄目前 table 中有的 Entry【記錄 10】。

VI. 實驗記錄

<table>
<tr><td>記錄</td><td colspan="2">內容</td></tr>
<tr><td>1</td><td colspan="2">開機訊息：</td></tr>
<tr><td rowspan="7">2</td><td>機器</td><td>IP address</td></tr>
<tr><td>H（host）</td><td></td></tr>
<tr><td>M（netmask）</td><td></td></tr>
<tr><td>R（router）</td><td></td></tr>
<tr><td>F（far）</td><td></td></tr>
<tr><td>N1（near1）</td><td></td></tr>
<tr><td>N2（near2）</td><td></td></tr>
<tr><td rowspan="4">3</td><td colspan="2">重開機後 ARP table 已有的 entry：</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">.
.
.
.</td></tr>
<tr><td rowspan="4">4</td><td colspan="2">ping 遠端機器（F）後 log 檔內容：
.
.
.
.</td></tr>
<tr><td rowspan="4">5</td><td colspan="2">ping 遠端機器（F）後 ARP table 現有 entry：</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">.
.
.
.</td></tr>
</table>

6	ping 近端機器（N1）後 log 檔內容： . . . .
7	ping 近端機器（N1）後 ARP table 現有 entry：
	. . . .
8	ping 近端機器（N1）後 log 檔內容： . . . .
9	ping 近端不存在或未上網機器（N2）後 log 檔內容： . . . .
10	ping 近端不存在或未上網機器（N2）後 ARP table 現有 entry：
	. . . .

Ⅶ. 問題與討論

1. 根據【記錄 1】，Trace 開機訊息的訊息（越多越好，尤其與網路相關的訊息），尋求每一訊息由哪一個檔案（核心程式碼或初始化 script）的哪一指令所產生，做成如下格式的表格。

開機訊息	檔案名稱	對應的指令或函式

. . . .	. . . .	. . . .

2. 請說明實驗小組 trace 網路卡驅動程式以及協定驅動程式的流程，及所用時間。

3. 分別解釋【記錄 3、4、5】（ping 遠端機器）、【記錄 6、7】（ping 近端機器）、【記錄 8、9】（ping 近端不存在或未上網機器）的結果，並綜合做出 ping 送出 ICMP echo 的流程圖。

4. 請自行設計一個問題，並自行回答。

Ⅷ. 參考文獻

[1] 交大資工 Linux 網站，linux.cs.nctu.edu.tw。可取得燒錄檔，或用 NFS、FTP 方式的安裝。

[2] Chinese GNU/Linux Extensions, http://cle.linux.org.tw, 中文化 linux 可開機光碟 image（燒錄檔）提供者。

[3] Alessandro Rubini, *"Linux Device Drivers,"* O'Reilly & Associations, Feb. 1998.

[4] 蔡品再、林盈達，追縱 Linux 核心的方法， http://speed.cis.nctu.edu.tw/~ydlin/miscpub/remote_debug.pdf

[5] Nemeth Snyder et al., *"UNIX System Administration Handbook,"* Third Edition, Prentice Hall, Aug. 2000.

[6] Linux kernel 的取得處，ftp://linux.cs.nctu.edu.tw/kernel。

[7] Douglas E. Comer, David L. Stevens *"Internetworking with TCP/IP Volume II,"* Third Edition, Prentice Hall, June 1998.

[8] Fedora Core download site, http://fedoraproject.org/wiki/.

[9] 鳥哥的私房菜：Linux 核心編譯與管理，http://linux.vbird.org。