# 10/15 嵌入式作業系統實習：

# The Linux Bootdisk

# 1

# 2 實習簡介

Linux 開機磁片 (boot disks) 在很多情況下是很有用的，諸如

- 測試一個新的核心 (kernel)。
- 從磁碟錯誤中復原 (這類錯誤從遺失開機磁區到磁碟讀寫頭毀損都有可能) 。
- 修復一個癱瘓 (disabled) 的系統
- 安全地升級臨界共用 (critical) 的系統檔案 (諸如 libc.so) 。

有幾種獲得 boot disks 的方法：

- 使用發行套件 (distribution) 像是 Slackware 所提供的。它至少能讓你開機。
- 使用救援套件 (package) 建造用來做爲救援磁片的磁片。
- 學習每一種 disk 運作系統時所需的東西，然後自己製作。

在此次實習中，我們將選擇最後一種方法，如此我們能靠自己動手做。這樣子，如果某處發生問題，就能找出辦法去解決問題。此外也可以學到很多有關 Linux 如何運作的知識。

# 3 實習目的 :(在 Linux 環境下，實際操作所附範例。)

建立一片 Bootdisk .

# 4 實習環境

Linux ; with kernel version 2.4 以上(含)

# 5 背景知識

## 4.1 Bootdisks and the boot process

尚未提及如何製作 Bootdisk 前，首先要瞭解整個系統開機的過程。

當啓動電源之際，電腦的 BIOS 就會找尋有無系統開機磁片，此時會有兩種狀況產生：

第一種狀況：找到系統開機磁片 --

此時就會從系統開機磁片中的第 0 磁區、第 0 磁柱載入可開機磁區

第二種狀況：找不到系統開機磁片 --

BIOS 就會找尋硬碟的 MBR (Master Boot Record)，並且執行記錄在 MBR 上的開機載入程式 (Boot Loader) 進行開機

無論是從軟碟開機也好，還是從硬碟開機也好，首先 OS Loader (就 Linux 來講就可能是 Lilo、Grub、syslinux) 會載入 Linux Kernel，而 Kernel 一旦起動後，第一件事就是切入保護模式 (protected mode)，此時，所有的硬體交由 Kernel 來控制，也意味掙脫 BIOS 的控制。

當 Kernel 載入完畢後，便開始初始化系統所有硬體設備。而當所有的硬體初始化的動作也告完成之際，系統將嘗試掛載 (mount) root filesystem。Root filesystem 就是被掛上當作 "/" 目錄的 filesystem (以下簡稱 fs)，當然，如果無法正確 mount 上 root fs，Linux 也只好 show 個訊息：

VFS: Unable to mount root fs on XXX

然後就會停止運作 (halt)，這邊的 XXX 是指那種 fs，這部份的訊息，可以參考 kernelsource 中 fs/super.c 的程式碼。

當 root filesystem 成功的掛載後，就會去執行 init 這個程式。init 會檢查 /etc/inittab，找出該檔中標明 sysinit 這行，並執行被指定的 script，在 Mandrake(or Redhat)上爲 /etc/rc.d/rc.sysinit，於是 rc.sysinit 肩負系統的初始化的大任，不外乎有以下任務：

. 呼叫 /sbin/initlog 紀錄系統初始化過程
. 設定 path、hostname 等資訊
. banner 畫面：一般我們看到 "Welcome to Mandrake Linux" 的訊息
. Mount /proc
. Load system font
. Configure kernel parameters
. Set the system clock
. Load keymap
. Start up swapping、turn on swap
. Remount the root filesystem read-write、Clear mtab

. Finding module dependencies、Load modules
. Check filesystems

另, /etc/rc.d/ 這個目錄包含一個複雜的子目錄架構(rc0.d~rc6.d)，其中的檔案指出如何啓動與關閉大部分的系統服務。

當 rc.sysinit 執行完畢，控制權立即移轉回 init 手中，進入預設 runlevel：

. 若內定的 runlevel 爲 3： 則 init 執行 /sbin/mingetty 啓動 virtual console，並且以 "login：" 提示讓使用者登入，以完成開機。登入後系統會提供一個 shell 給使用者，就可以使用 Linux 。
. 若 runlevel 爲 5：則在開啓 virtual console 後，init 會再執行 xdm 啓動 X window system，讓使用者以 xdm 界面登入。

以上就是 Linux 開機的過程。

## 4.2 BusyBox

BusyBox 它包含了七十多種 Linux 上標準的工具程式，只需要的磁碟空間僅僅幾百 k (視所選擇工具程式的數目來決定大小)，在嵌入式系統上常用到它，可在 http://www.busybox.net/ 找到參考資料及下載。

而 Busybox 爲何能夠佔有如此小的容量，卻能提供爲數不少的常用工具呢？這訣竅在於 busybox 在編譯後，雖然整體只是一個執行檔，卻可以透過 symbolic link 的方式，將常用指令 (Busybox 的術語是 "applet") 「連」到 busybox 這個執行檔上，如：# tree bin/

```
|-- ash
|-- busybox
|-- cat -> busybox
|-- chgrp -> busybox
|-- chmod -> busybox
|-- chown -> busybox
|-- cp -> busybox
|-- date -> busybox
|-- df -> busybox
|-- dmesg -> busybox
|-- echo -> busybox
|-- false -> busybox
|-- fdflush -> busybox
```

```
|-- grep -> busybox

[後略]
```

如我們所見，cat、chgrp、chown、... 都被 symbolic link 到 busybox，那，要如何運作呢？

咱們翻開 busybox 的 source 片段來看：(busybox.c)

```
int main(int argc, char **argv)
{
    const char *s;

    for ( s = applet_name = argv[0]; *s != '\0'; )
    {
        if ( *s++ == '/' )
        applet_name = s;            /* 取得 applet 名稱 */
    }

    run_applet_by_name( applet_name, argc, argv );
    /* applet 進入點 */
    error_msg_and_die( "applet not found" );
}
```

main(int argc, char **argv) 的引數中，argv[0] 是什麼呢？就是「執行時期的名稱」，換言之，就是 cp、cat、chown 等等，確定使用者所下的指令後，就丟給 run_applet_by_name() 這個執行引擎去跑，示意圖如下：

```
something(我們的指令) ---------|
                                |                cp
    buzybox <-------------------|                /
                                               /---mv
1. 辨識 applet 名稱                            |
2. 丟進 run_applet_by_name() 去跑 ===> +-mkdir
```

透過功能共用的方式，不僅可以共享相似的副程式 (function)，更減少重複 link 的負擔，於是乎，不需要替每個指令都製造新的執行檔，只需要在 busybox 中撰寫 applet 的功能即可，大幅縮小空間佔有量。

# 6 實習步驟

在此所製作的簡易版 Bootdisk，是一個功能非常精簡的 Linux，只支援軟碟、並不支援硬碟及光碟，沒有網路功能。可以執行簡單的 shell 程式及一些常用的工具程式。

## 6.1 作業環境 & Tools

接下來，你應該準備一些發展 Bootdisc 的 source package 及工具程式有：

. Linux Kernel source code : v 2.6.6
**ftp://linux.sinica.edu.tw/pub1/kernel/v2.6/linux-2.6.6.tar.gz**

. BuzyBox : busybox-1.00-rc3.tar.gz **http://www.busybox.net/**
[匯集常用指令於單一執行檔的工具集 ; a Swiss knife]

. syslinux :基本上現在都有內建
[Linux Kernel Loader，可以讀取 FAT，但不支援 Ext2; 因 size 很小:only 4kb，適合製作 bootdisk 時使用，]

Bootdisc 的開機過程與一般開機過程稍有不同：將 LILO、GRUB 更換成 syslinux、開機完之後直接提供一個 shell 給使用者用，其完整開機方式如下 ：

BIOS --> syslinux --> kernel --> init --> shell

## 6.2 build kernel

首先，必須針對我們的 Bootdisc 量身製作它所屬 kernel，最好也把所需的 driver make 進核心中，如下：
(將 kernel source code 置於 /usr/src/ 底下.如果不想 Build kernel 可以到/boot 底下去找 vmlinuz-2.4.20 這是 redhat9.0 的 bzImage)

```
# make menuconfig
# make dep
# make bzImage
# make modules
# make modules_install
```

Kernel 建構完畢後，

(通常 bzImage 會位於 ./arch/i386/boot/bzImage)我們還需要一些常用的工具程式，兼具便利與低容量使用，Busybox 是個不錯的選擇。

## 6.3 **Root Filesysem**

進入打造迷你的 root fs 的步驟。在開始建造 root fs 之前，必須以 root loggin，因爲接下來必須要用到 mknod。

首先，爲 root fs 建一個目錄叫做 temp，進入 temp：

```
# mkdir /temp
# cd /temp
```

(注意 temp 是完全空的以下的東西都是在 temp 下建出來)

再來爲 root filesystem 建立一些標準的目錄：

```
# mkdir  dev  etc  etc/rc.d  bin  proc  mnt  tmp  var
# chmod 755  dev  etc  etc/rc.d  bin  mnt  tmp  var
# chmod 555  proc
# ln -s  bin sbin
```

進入 /dev 目錄下建立一般終端機設備：

```
# cd dev
# mknod  tty  c  5  0
# mknod  console  c  5  1
# chmod 666  tty  console
```

接著建立 VGA Display 虛擬終端機設備：

```
# mknod  tty0  c  4  0
# chmod 666  tty0
```

再建立 RAM disk 設備：

```
# mknod ram0  b  1  0
# chmod 600 ram0
```

建立 floppy 設備：

```
# mknod   fd0   b   2   0
# chmod   600   fd0
```

最後再建立 null 設備：

```
# mknod null   c   1   3
# chmod 666   null
```

請進入到 /temp/etc/rc.d 這個目錄下編輯 inittab (vi inittab)，內容如下：

```
::sysinit:/etc/rc.d/rc.sysinit
::askfirst:/bin/sh
```

修改 inittab 的權限：

```
# chmod 644 inittab
```

編輯好 inittab 之後，緊接著就是編輯 /temp/etc/rc.d 底下的 rc.sysinit (vi rc.sysinit)
如下：

```
#!/bin/sh
mount -a
```

變更其權限：

```
# chmod 755 rc.sysinit
```

再編輯 /temp/etc/fstab，(vi fastab)
fstab 內容如下：

```
proc    /proc       proc       defaults     0     0
```

修改 fstab 權限：

```
# chmod 644 fstab
```

**Busybox**

接著趕緊建立出一個靜態連結的 BusyBox (init, getty，login, mount 這些指令是一定要的)，之所以要編譯成靜態連結的原因，就是不希望 Floppy Linux 使用到 glibc 而增加磁碟的使用空間。擁有 fs 的框架後，現在開始編輯有關的 shell script。我們先從/etc/inittab 這一支 script 下

手，因爲我們用的是 BusyBox 上的 init，與一般所使用的 init 不太一樣，會先執行 /etc/init.d/rcS 而非 /etc/rc.d/rc.sysinit，爲了做出來的 Bootdisc 架構與 Mandrake/Redhat
架構一樣，所以修改了 BusyBox 中的 init.c 。(在 Busybox 下的 init 這資料夾)

底下是修改的內容：(找這段並修改)

```
#ifndef INIT_SCRIPT
#define INIT_SRCIPT "/etc/rc.d/rc.sysinit"
#endif
```

以下爲建立 BusyBox 的步驟：

```
# tar -zxvf busybox-1.00-rc3.tar.gz
# make menuconfig   (選要用的指令)
```

( 這兩各要勾 ：

1.General Configuration
→support --install [-s] to install applet links at runtime.
勾這各 busybox 會自動幫你做 link..不然要自己做..指令多就很慘

2. Build Options
→Build Busybox as a static binary)

```
# make     (記住要先改 init.c 喔)
# make install
```

完成上述編輯之後，就要把靜態連結版的 BusyBox 搬到 /temp/bin/
基本原理可看背景知識..

```
# cp –d   busybox-1.00-rc3/_install/bin/* /temp/bin
# cp –d   busybox-1.00-rc3/_install/sbin/* /temp/bin
# cp –d   busybox-1.00-rc3/_install/sbin/linuxrc /temp/
```

(-d 爲保留 symlink 的參數)

## 5.5 Ramdisk

到這裡爲止，可以說 temp 的 root fs 已經製作完畢，不過，一般來說，我們會採取 RAM Disk 的方式實現。爲什麼呢？如果我們採取 ramdisk 的方式，而非實體 (physical) 存在於儲存媒體 (當然這裡是

指 floppy) 中，那麼，我們甚至可以進一步壓縮這個 ramdisk image，以便放置更多附加功能。

現在，我們就來製作 ramdisk ，其方法如下(Mydisk 是我在根目錄下建的目錄,ramdisk 不用建打下面指令會自然產生

```
# dd if=/dev/zero of=/Mydisk/ramdisk bs=1k count=2048
# losetup /dev/loop0 /Mydisk/ramdisk
# mke2fs -m 0 /dev/loop0
# mount -t ext2 /dev/loop0 /Mydisk/
# cp -a /temp   /Mydisk
# umount /dev/loop0
 # losetup -d /dev/loop0
# dd if=/Mydisk/ramdisk | gzip -9 > /Mydisk/Image.gz
```

註在此可以利用 du 指令來看檔案 size

#du Mydisk/Image.gz 看是否有壓成..並注意最好不要超過 330 以上.不然有可能塞不進去磁片因 kernel 佔了很大的空間.當然你也可將 kernel 編小一點就可不必擔心這各問題

```
# sync
```

此時,可以在 /Mydisk 底下發現 Image.gz 這個檔案,這就是 ramdisk 影像檔，當然啦，假如經常修改 root fs 的內容，大可將上述步驟寫成一支 shell srcipt，省下無謂的時間浪費。

到目前爲止，我們已經完成製作 Bootdisc 的準備，接下來，就將我們努力的成果擺入 floppy。Bootdisc 的 loader 爲 syslinux。

## 5.6 Syslinux

首先，將空白的磁片格式化，然後載入 sysliunux，步驟如下：

```
# mkdosfs /dev/fd0   (也可藉 windows format floppy)
# syslinux /dev/fd0
```

完成上述步驟之後，請編輯 syslinux 的組態檔 syslinux.cfg，其 syslinux.cfg 內容如下：

```
TIMEOUT 20
DEFAULT bzImage
```

LABEL bzImage
KERNEL bzImage
APPEND root=/dev/ram0 initrd=Image.gz

(詳細選項說明可見 syslinux 中 README ; **http://syslinux.zytor.com/**)

將 syslinux.cfg、kernel、Image.gz 拷貝到磁片中：

```
# mount   /dev/fd0   /mnt/floppy
# cp   /usr/src/linux/arch/i386/boot/bzImage   /mnt/floppy
# cp   /Mydisk/Image.gz   /mnt/floppy
# cp   syslinux.cfg   /mnt/floppy
```

以上即完成 Linux Bootdisk.　請重開機測試是否可 boot system !!

## 7　結論:

這實習的目的主要是讓同學能藉由 **ramdisk** 來了解一各 **kernel** 要 **boot** 起來至少須哪些東西.如 **filesystem.**一些必備程式**(init…**等等**).**還有熟悉 **busybox** 這各好用的東西**.**

## 8　問題與討論

1． busybox支援了哪些指令．

2． ramdisk主要的用途．如用在哪**?**好在哪？

## 9　*Reference*：


[1] http://www.tldp.org/HOWTO/Bootdisk-HOWTO/

[2] http://www.busybox.net/ , the main busybox website

[3] http://www.uclibc.org/, Uclibc – the compact C library

[4] http://www.linuxinfor.com/english/Remote-Serial-Console/configure-boot-loader-syslinux.html, online linux source