The Linux Bootdisk HOWTO

作者: Tom Fawcett (*fawcett+BH@croftj.net*)

譯者: 朱漢農

v4.0, April 2000, 翻譯日期: 27 July 2000

本文描述如何設計與建造你自己的 Linux boot/root 磁片。這些磁片能用來當做救援磁片 (rescue disks)，或是能用來測試新系統元件 (components)。在企圖建造你自己的 bootdisk 之前，你應該要相當熟悉系統管理工作。如果你只是想要一張緊急時使用的救援磁片，請參考 A.1 (Pre-made bootdisks)。

目錄

1 前言 – Preface

這份文件可能已經過期了。 如果標題頁上的日期距今已超過 6 個月，那麼請查閱 *Bootdisk-HOWTO homepage* `<http://www.croftj.net/~fawcett/Bootdisk-HOWTO/index.html>` 看看是否已有較新的版本。

雖然本文以 text 格式也是可以閱讀，但是因爲印刷符號的關係， 最好 還是以 Postscript (.ps)、 PDF 或 HTML 的格式來閱讀。

1.1 版本注意事項 – Version notes

Graham Chapman 是原本 Bootdisk-HOWTO 的作者，他一直提供支援到 version 3.1。 Tom Fawcett 大約是在 kernel v2 問市時成爲合作作者。他是本文目前的維護者。

文中資訊是給在 **Intel** 平台上運作的 Linux 使用。其中許多資訊也許能應用在其它平台的 Linux，但是我們並沒有嘗試在其它平台製作 bootdisk，也沒有相關的資訊。如果你有在其它平台上製作 bootdisk 的經驗，請與我們聯絡。

1.2 尚未完成的事

有任何自願者嗎？

1. 請描述 (或是鏈結到另一份有敘述的文件)如何製作其它可開機的類磁片物品，諸如 CDROMs, ZIP disks 與 LS110 disks。
2. 請描述如何處理巨大的 `libc.so` 共享函式庫。基本上可選擇獲取較舊、較小的函式庫，或是刪減現有的函式庫。
3. 重新分析 distribution bootdisks 與更新 "How the Pros do it" 這一節。
4. 刪減敘述關於如何升級現有 distribution bootdisks 的章節。 This is usually more trouble than it's worth.
5. 重寫 / 潤飾 Troubleshooting 這一節。

1.3 回應與感謝 – Feedback and credits

我接受任何關於本文內容之回應，無論是好是壞。我/我們 已力求這份文件內的指令與資訊是正確而可靠的。如果你發現任何錯誤或遺漏，請讓我知道。在撰寫時， 請指出你所參考的文件之版本號碼 。

我們感謝許多協助修正與給予建議之人。他們的貢獻使得本文比我們自己獨立完成它時還來得更好。

請各位利用上述的 email 地址，給予作者你的批評、指正與疑問。我不介意嘗試回答任何問題，但是如果你有特定問題是關於你的 bootdisk 不能運作，那麼請先閱讀 7 (Troubleshooting)。

1.4 散佈政策 – Distribution policy

2 簡介 – Introduction

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`)。

有好幾種獲得 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。

有些人選擇最後一種方法，如此他們能靠自己動手做。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。

本文假設讀者已熟悉基本的 Linux 系統管理觀念。舉例來說，你應該知道有關目錄、 filesystems 與軟碟片的議題。你也應該知道如何使用 `mount` 與 `df`。你還應該知道 `/etc/passwd` 與 `fstab` 這兩個檔案的用途以及它們看起來像什麼。最後，你應該知道 HOWTO 文件內大部分的指令，都要以 root 的身份來執行。

剛開始製作你自己的 bootdisk 是很複雜的。如果你未曾讀過 Linux FAQ 與相關文件，諸如 Linux Installation HOWTO 與 Linux Installation Guide，那麼你不應該嘗試建造開機磁片。如果你只需要緊急時用的 bootdisk，下載一個別人事先完成的 bootdisk 拿來用會 更 爲容易。請參考下面的 A.1 (Pre-made bootdisks) 以得知在哪裡可以找到這些東西。

3 Bootdisks與開機流程 – Bootdisks and the boot process

bootdisk 基本上是放在軟碟片內的一個小型而自足的 Linux 系統。它必須執行許多和完整又 full-size 的 Linux 系統相同的功能。在建造 bootdisk 之前，你應該了解基本的 Linux 開機流程。我們在此只做基本的介紹，但已足夠讓你了解本文之後的內容。很多細節與替代選項已被省略。

3.1 開機流程 – The boot process

所有 PC 系統開始開機流程都是藉由執行 ROM (明確地說，就是 BIOS) 中的程式，從開機磁碟機的第 0 磁區、第 0 磁柱載入可供開機的磁區。開機磁碟機通常是第一台軟碟機 (如 DOS 的 `A`槽 與 Linux 的 `/dev/fd0`)。接著 BIOS 會嘗試執行這個磁區。在大部分可開機的 disks 上，第 0 磁區、第 0 磁柱包含以下兩者之一：

- 開機載入程式 (boot loader，如 LILO)的程式碼，它會找出 kernel 所在位置，接著載入並執行它以啟動開機程序。

- 一個作業系統 kernel 的開頭 (start)，諸如 Linux。

如果一個 Linux kernel 已利用 raw-copied 的方式置入一張磁片內，那麼這張磁片的第一個磁區就是 Linux kernel 本身的第一個磁區。這個磁區將從開機設備載入 kernel 的剩餘部分以繼續開機流程。

旦 kernel 載入完畢，一些基本設備也完成初始化 (initialization)。然後系統將嘗試從某個設備載入以及掛上 (mount) **root filesystem**。所謂的 root filesystem 只是一個被掛上當作 "/" 目錄的 filesystem。 kernel 必須被告知可從哪裡找到此 root filesystem；如果 kernel 在那裡找不到一個可載入的影像檔 (image)，系統就會停止運作 (halt)。

在某些開機情況下 – 常常是從軟碟片開機 – root filesystem 會被載入到 **ramdisk** 中，也就是被系統所存取的 RAM，如同系統存取磁碟一般。爲何系統會載入到 ramdisk 的理由有二。第一， RAM 是幾個比軟碟片快的有序磁性體，所以系統在其上運作較快； 第二， kernel 可以從軟碟片載入一個 壓縮的 **filesystem** 並且在解壓縮後放到 ramdisk 上，如此可讓更多的檔案儲存在軟碟片上。

一旦 root filesystem 被載入並掛上，你會看到一行訊息像：

```
VFS: Mounted root (ext2 filesystem) readonly.
```

此時系統會在 root filesystem 上找到 `init` 程式 (在 `/bin` or `/sbin`) 並執行它。 `init` 讀取它的組態設定檔 (configuration file) `/etc/inittab`，找出檔中標明 `sysinit` 的一行，並執行被指名的script。這個 `sysinit` script 通常類似 `/etc/rc` 或 `/etc/init.d/boot` 這兩個檔。這個 script 是一組建立基本系統服務的 shell 指令，諸如：

- 對所有磁碟執行 `fsck`,
- 載入必備的核心模組 (modules),
- 啓動 swapping,
- 進行網路初始化,
- 將指定在 `fstab` 內的磁碟掛上。

這個 script 常會啓動其它各種不同的 scripts 執行模組的 (modular) 初始化。舉例來說，在一般的 SysVinit 架構下， `/etc/rc.d/` 這個目錄包含一個複雜的子目錄架構，其中的檔案指出如何啓動與關閉大部分的系統服務。然而，在一張 bootdisk 上，這樣的 sysinit script 常常是非常簡單的。

當 sysinit script 結束後，控制權回到 `init` 上，接著進入 預設的*runlevel*，此預設的 runlevel 以 `initdefault` 這個關鍵字被指定在 `inittab` 內。此 runlevel line 通常指定一個像 `getty` 的程式，這個程式負責處理 console 與 ttys 之間的通訊。事實上，就是 `getty` 程式在螢幕上印出熟悉的 "`login:`" 提示。 `getty` 程式並轉而呼叫 `login` 程式以處理 login 是否有效，並於之後建立 user sessions。

3.2 磁碟類型 – Disk types

如果你已了解基本的開機流程，那麼我們現在可以定義所涉及到之各種不同類型的磁碟。我們將磁碟分類成四種。本文與在此討論所稱之 "磁碟 (disk)" 除非有特別聲明，否則都是指軟碟片，雖然絕大部分的討論也同樣可應用在硬碟上。

boot

一張包含可被啓動之 kernel 的磁片。這張磁片被用來啓動 kernel，接著這個 kernel 會載入在另一張磁片上的 root file system。在 bootdisk 上的 kernel 通常必須被告知到哪去找它的 root filesystem。

bootdisk 常會從另一張磁片載入一個 root filesystem，但是相反地， bootdisk 也有可能被設定成載入硬碟的 root filesystem。一般在測試新 kernel 時會這樣做 (事實上，“`make zdisk`” 會自動地從 kernel 的原始碼製造出這樣的一張 bootdisk。

root

在其 filesystem 上包含運作 Linux 系統必備檔案的一張磁片。這張磁片不一定有 kernel 或 boot loader。

一旦 kernel 被啓動後， root disk 就可以獨立於其它磁片來運作系統。通常 root disk 的內容會被自動地 copy 到 RAM 而成爲 ramdisk。這使得 root disk 的存取變得更爲快速，而且可釋放一台軟碟機給工具程式磁片 (utility disk)。

boot/root

一張同時包含 kernel 與 root filesystem 的磁片。換句話說，這張磁片包含不用硬碟而能啓動與運作 Linux 系統之所有必備項目。這種磁片的優點在於簡單輕便 – 每一項必備的東西都放在同一張磁片上。然而，隨著檔案 size 的逐漸增加，讓所有東西都存在同一張磁片上就越顯困難，甚至利用壓縮也一樣。

utility

一張包含 filesystem 的磁片，但是並不是要掛上做爲 root file system 來使用。這張磁片可視爲額外的資料片 (data disk)。你可以利用這種磁片把原本過多而不能放在 root disk 上的工具程式放在其上。

一般而言，當我們提及“建造一張 bootdisk” 時，是指造出 boot (kernel) 與 root (files) 這兩個部分。這兩個部分不是放在一起 (一張單張的boot/root disk)，就是兩張分開的磁片 (boot + root disks)。對救援磁片而言最具彈性之做法，可能是使用兩張分開的 boot 與 root 磁片，再加上一張或多張的 utility diskettes 以處理多出來的東西。

4 建立一個root filesystem – Building a root filesystem

造出 root filesystem 涉及選擇能讓系統正常運作所必備的檔案。在這一節中，我們將敘述如何建造一個 壓縮的 *root filesystem*。在磁片上建造一個直接掛上做爲根目錄 (root) 之未經壓縮的 filesystem 是較不普遍的觀念；這個替代方案敘述在 8.2 (Non-ramdisk Root Filesystem) 這一節中。

4.1 概觀 – Overview

root filesystem 必須包含支援完整 Linux 系統運作所需的每一個項目。爲了能夠達成這個目的，這張磁片必須包括能讓 Linux 系統運作最起碼 (minimum) 的需求：

- 基本的檔案系統架構，
- 最起碼的目錄： `/dev`, `/proc`, `/bin`, `/etc`, `/lib`, `/usr`, `/tmp`,
- 基本的工具程式： `sh`, `ls`, `cp`, `mv`, etc.,
- 最起碼的組態設定檔： `rc`, `inittab`, `fstab`, etc.,

- 設備檔： /dev/hd*, /dev/tty*, /dev/fd0, etc.,
- Runtime 函式庫以提供工具程式所使用之基本功能 (functions)。

當然，任何系統只有在你能於其上執行某些東西時才會顯得有用，而一張 root 磁片通常只有在你能做到以下事情時才會顯得有用：

- 檢查另一台磁碟機的 file system，舉例來說，檢查你硬碟上的 root file system，你必須能夠從另一台磁碟機啓動 Linux，例如你可以用一張 root 磁片辦到這件事。然後你可以在你原本的 root 磁碟機未被掛上時，對其執行 fsck 。
- 使用檔案 (archive) 與壓縮工具程式，諸如 cpio, tar, gzip 與 ftape，從備份 (backup) 恢復儲存所有或部分你原本 root 磁碟機的資料。

我們將敘述如何建造一個 壓縮的 filesystem，就是平時被壓縮在磁片上，只有當開機時，才會解壓縮後存入ramdisk。 用壓縮 filesystem 的方式，你可以在一張標準的 1440K 磁片上放入很多檔案 (大約 6 megabytes)。因為 filesystem 比磁片大很多，我們不能直接把它建在磁片上。我們必須在其它地方建立它，壓縮它，然後再把它 copy 到磁片上。

4.2 製作 filesystem – Creating the filesystem

為了建造如此的一個 filesystem，你需要一個多出而夠大的設備，能夠讓你在壓縮之前存放所有的檔案。你將需要一個能夠存放大約 4 megabytes檔案的設備。有以下幾種選擇：

- 使用 **ramdisk** (DEVICE = /dev/ram0)。在這種情況下，記憶體被模擬成一台磁碟機。 Ramdisk 必須大到能夠存放一個適當大小的 filesystem。如果你使用 LILO，請檢查你的組態設定檔 (/etc/lilo.conf)，找到一行像

  ```
  RAMDISK = nnn
  ```

  這行決定可以分配給 ramdisk 的 RAM 之極大值。預設值是 4096K，這應該是足夠了。你應該不可能嘗試在一台少於 8MB RAM 的電腦上使用如此的 ramdisk。

  請檢查以確認你有一個設備檔像是 /dev/ram0,/dev/ram 或是 /dev/ramdisk。如果沒有，請自己以 mknod (major number 1, minor 0)造出 /dev/ram0。

- 如果你有一個未使用且夠大的硬碟 partition (差不多幾megabytes大就可以了)，就使用它吧。
- 使用一個 **loopback device**，這可以把一個磁碟檔案當做是一台設備來使用。使用 loopback device 時，你可以在硬碟上造出一個 3 megabyte 的檔案，並於其上建造 filesystem。

  鍵入 man losetup 找尋指令以使用 loopback devices。如果你沒有 losetup，你可以從 <ftp://ftp.win.tue.nl/pub/linux/utils/util-linux/> 目錄中， util-linux 套件 (package) 內相容版本之 mount 與 unmount 的隨附物中找到它。

  如果在你的系統上沒有 loop device 檔 (/dev/loop0, /dev/loop1, etc.)，那麼你必須用 “mknod /dev/loop0 b 7 0” 自己造出一個。一旦安裝好這些特別的 mount 與 umount 二進位檔，就請在一台容量夠大的硬碟上造出一個暫存檔 (temporary file)(eg, /tmp/fsfile)。你可以使用像這樣子的指令：

  ```
  dd if=/dev/zero of=/tmp/fsfile bs=1k count=nnn
  ```

以造出一個 *nnn*-block 的檔案。

請使用自己的檔名取代以下的 DEVICE。當你下了 mount 指令，你同時要加上“`-o loop`”選項以告知 mount 是使用 loopback device。 舉例來說：

```
mount -o loop -t ext2 /tmp/fsfile /mnt
```

以掛上 loopback device 的方式，把 `/tmp/fsfile` 掛上 `/mnt` 這個 mount point。用 `df` 指令可讓你看到以上的結果。

在你選擇其中一種方法後，請準備 DEVICE 以：

```
dd if=/dev/zero of=DEVICE bs=1k count=4096
```

這行指令送出一堆 0 把DEVICE填滿。用 0 填滿 device 是關鍵的一步，因爲 filesystem之後將會被壓縮，所以所有未使用的部分應被用 0 填滿以達到最大的壓縮比。無論何時你從你的 root filesystem 刪除檔案，請記得這個事實。實際上 filesystem 只是釋出 (de-allocate) 這些 blocks， 但是並沒有再把它們填爲 *0*。如果你執行過很多次刪除與 copy 的動作，你的壓縮 filesystem 最後會比必要的大出很多。

下一步就是造出 filesystem。 Linux kernel 承認兩種能讓 root disks 自動地被 copy 到 ramdisk 上的 file system。它們是 minix 與 ext2，其中 ext2 是比較受歡迎的。如果使用 ext2，你會發現使用 `-i` 選項指定比預設值更多的 inodes 是有助益的；我們建議用 `-i 2000`，這樣你就不會用完 inodes。如果不用上述選項，你可以移除許多不必要的 `/dev` 檔案以節省 inodes。 `mke2fs` 預設會造出 360 個 inodes在一張 1.44Mb 的磁片上。我發現在我目前的救援 root 磁片上， 120 個 inodes 是相當足夠了，但是如果你把所有的設備檔都放入 `/dev` 目錄中，那麼你很容易會超過 360 個inodes。使用壓縮的 root filesystem 可讓你擁有較大的 filesystem，同時預設會有更多的 inodes，但是你仍然必須要不就是減少檔案數量，要不就是增加 inodes 數目。

因此，你所使用的指令看起來會像這樣：

```
mke2fs -m 0 -i 2000 DEVICE
```

(如果你使用的是一個 loopback device，那麼請用你目前所用的磁碟檔案替換掉上面的 DEVICE。)

`mke2fs` 指令會自動地偵測可獲得的空間，然後依據偵測對自身進行組態設定。“`-m 0`” 參數避免保留空間給 root，因此可提供更多可用的磁碟空間。

下一步，掛上這個 device：

```
mount -t ext2 DEVICE /mnt
```

(如果 mount point 並不存在，你必須自行造出一個 mount point `/mnt`。) 在剩下的章節中，所有的目的 (destination) 目錄都被假設是相對於 `/mnt`。

4.3 移植檔案系統 – Populating the filesystem

以下是你的 root filesystem 最起碼該有的目錄

此處所呈現之目錄架構僅供 root diskette 使用。眞正的 Linux 系統有一套更爲複雜且設計良好的架構方法，稱爲 *Filesystem Hierarchy Standard*，它決定檔案應該如何放置。：

- /dev – 裡面存放設備檔 (Devices)，爲達成 I/O 工作所需
- /proc – Directory stub required by the proc filesystem
- /etc – 裡面存放系統組態設定檔
- /sbin – 重要的 (critical) 系統二進位執行檔 (binaries)
- /bin – 被認爲是系統一部分的基本二進位執行檔
- /lib – 提供 run-time 支援的共享函式庫
- /mnt – 維護其它磁碟所用的磁碟掛入點 (mount point)
- /usr – 額外的工具程式與應用程式

上述目錄的其中三個在 root filesystem 上會是空的，所以它們只需要用 mkdir 造出來。 /proc 目錄基本上是一個把 proc filesystem 放置於其下的 stub。 /mnt 與 /usr 這兩個目錄只是在 boot/root 系統運作時所使用的 mount points。因此再重覆一次，這些目錄只需要被造出來就可以了。

剩下的四個目錄描述於以下各節。

4.3.1 /dev

/dev 目錄包含一群特別的檔案，這些檔案是給系統上所有設備使用的，這樣的 /dev 目錄每個 Linux 系統都一定會有。這個目錄本身是一個普通目錄，可以以一般的方法用 mkdir 造出來。然而，這些特別的檔案必須以特別的方法用 mknod 指令造出來。

但還是有一條捷徑 – 直接 copy 你現有 /dev 目錄的內容，然後再清除你不想要的設備檔。唯一的要求是 copy 這些特別的設備檔時，要用 -R 選項。這個選項會 copy 整個目錄中的檔案，但是不會 copy 這些檔案的內容。請確定使用 大寫字母 *R*。這個指令是：

```
cp -dpR /dev /mnt
```

在此我們假設磁片是被掛在 /mnt 底下。 dp 選項 (switches) 確保 symbolic links 是以 links 的方式來 copy，而不是 copy 鏈結檔所指向的 target file，同時原本的檔案屬性也被保留，因此保留了檔案的所有權資訊。

如果你想要用高難度技巧完成這個任務，就利用 ls -l 列出你想要的設備檔之 major 與 minor device numbers，然後再用 mknod 在磁片上造出它們。

無論如何 copy 這些設備檔，還是要檢查任何你所需之設備檔 (device special file) 是否已放入這張救援磁片中。舉例來說， ftape 使用磁帶設備，如果你想要從 bootdisk 存取軟式磁帶機，你就需要 copy 所有有關的設備檔。

請注意，每一個設備檔需要一個 inode，但 inodes 一直都是稀少的資源，特別是在磁片 filesystems 上。因此，從磁片上的 /dev 目錄移除任何你所不需要的設備檔是有意義的。舉例來說，如果你沒有 SCSI 磁碟，你可以放心地移除所有以 sd 開頭的設備檔。同樣地，如果你並不想使用你的序列埠 (serial port)，那麼你也可以移除所有以 cua 開頭的設備檔。

請確定從這個目錄放入了以下檔案的： console, kmem, mem, null, ram0 and tty1.

4.3.2 /etc

這個目錄包含了重要的組態設定檔。在大部分的系統上，這些檔案被分爲三個群組：

1. 一直都是必備的， *e.g.* `rc`，`fstab`，`passwd`。
2. 可能是必備的，但是沒有人能十分確定。
3. 偷跑進來的垃報。

通常可以用以下指令識出哪些是非基本的檔案：

```
ls -ltru
```

這個指令將檔案依據上次被存取的日期，以先早後晚 (reverse) 的順序列出，所以如果有任何檔案不會被存取，那麼它們就可以從 root 磁片中刪去。

在我的 root 磁片上，我的組態檔數目已減至 15 個。這可減少我處理以下三種檔案的工作：

1. 我必須爲 boot/root 系統進行組態設定的檔案：
   (a) `rc.d/*` – 系統啓動與改變 run level 的 scripts
   (b) `fstab` – 要被掛上的 file systems 清單
   (c) `inittab` – 給 `init` process 的參數，於開機時啓動的第一個 process。
2. 我們應該爲 boot/root 系統整理的檔案：
   (a) `passwd` – 重要的使用者、 home 目錄等其它項目的清單。
   (b) `group` – 使用者群組。
   (c) `shadow` – 使用者的密碼。你可能沒有這個檔。
   (d) `termcap` – the terminal capability database.

   如果系統安全 (security) 對你很重要，那麼 `passwd` 與 `shadow` 應該被刪減，以避免將使用者密碼 copy 出系統，這樣當你從磁片開機時，不想要的 logins 會被拒絕。

   請確定 `passwd` 至少包含了 `root`。如果你要讓其他的使用者 login，請確定他們的 home 目錄與 shells 都存在。

   `termcap`，終端機資料庫，一般而言有幾百個 kilobytes。你 boot/root 磁片的版本應該被刪減到只包含你所使用的終端機，這通常就是 `linux` 或 `linux-console` 項目 (entry)。
3. The rest. They work at the moment, so I leave them alone.

Out of this, 我實際上只必須設定兩個檔，而它們所應包含的項目驚人地少。

- `rc` 應該包含：

```
#!/bin/sh
/bin/mount -av
/bin/hostname Kangaroo
```

請確定上述的目錄都是正確的。你並不需要眞地去執行 hostname – 如果你執行只是讓系統比較好看 (譯註：如此系統會有個名字)。

- fstab 應該至少要包含：

```
/dev/ram0       /          ext2   defaults
/dev/fd0        /          ext2   defaults
/proc           /proc      proc   defaults
```

你可以從你現存的 fstab copy 你想要的項目，但是你並不應該自動地掛上你硬碟任何的 partitions；請對這些項目使用 noauto 關鍵字 (譯註：用 noauto 代替 default)。當使用 bootdisk 時，你的硬碟可能是早已損壞或掛了。

你的 inittab 應該被改變，以使其中 sysinit 這行能執行 rc 或無論什麼將被執行的基本開機 script。同時，如果你想要確保不可從序列埠 login，請在所有行尾包括 ttys 或 ttyS 的 getty 項目前加上「#」符號 (comment out)。請保留 tty 埠以讓你可以在 console 前 login。

一個最起碼的 inittab 檔看起來樣這樣：

```
id:2:initdefault:
si::sysinit:/etc/rc
1:2345:respawn:/sbin/getty 9600 tty1
2:23:respawn:/sbin/getty 9600 tty2
```

inittab 檔定義了系統在各種不同的情況中將執行什麼項目，包括 startup、切換至多使用者模式等等。請仔細地檢查在 inittab 中被提及的檔案名稱 (filenames)；如果 init 不能找到所提及的程式，那麼 bootdisk 將會停止運作，而你甚至不會得到錯誤訊息。

請注意，某些程式不能被移到其它地方，因爲其它程式已在撰寫時，就把它們的檔案位置寫死了 (hardcode)。舉例來說在我的系統上， /etc/shutdown 已把 /etc/reboot 的位置寫死在其中。如果我移動了 reboot 到 /bin/reboot，然後下達一個 shutdown 指令，將會因爲找不到 reboot 檔而發生錯誤。

剩下來的，就是 copy 在你 /etc 目錄中的所有文字檔 (text files)，再加上在你 /etc 目錄中，你無法確定你需不需要的所有可執行檔。需要指示 (guide) 者，請參考在 C (Sample roodisk directory listings) 的樣本清單。也許只要 copy 這些檔案就足夠了，但是系統差異會有很大的影響，所以你無法確定你系統上相同的檔案組合，就一定等於清單中的檔案。唯一確定的方法就是從 inittab 著手，並找出需要什麼。

現在大部分的系統使用 /etc/rc.d/ 目錄，其中包含給不同 run levels 的shell scripts。最起碼會有一個單一的 rc script，但是僅從你現存的系統 copy inittab 與 /etc/rc.d 這兩個目錄，然後刪減 rc.d 目錄中的 shell scripts 以移除和磁片系統環境無關的 processing，會是較爲簡單的做法。

4.3.3 /bin 與 /sbin

/bin 目錄是一個放置爲了執行基本作業 (operations) 而所需之額外工具程式的方便好地方，這些工具程式諸如 ls, mv, cat 與 dd。 bin/ 與 /sbin 這兩個目錄的檔案清單範例請見 C (Sample rootdisk directory listings)。但範例中並沒有包括任何從備份復原時所需之工具程式，諸如 cpio, tar 與 gzip。這是因爲我把這些東西放在另一張 (separate) 工具程式磁片上，以節省 boot/root 磁片的空間。一旦 boot/root 磁片被開機啓動，就會被 copy 到 ramdisk 並釋放軟碟機，讓軟碟機能掛上另一張磁片，就是工具程式片。我通常把它掛上當做 /usr。

工具程式磁片 (utility diskette) 的製作被描述在下面 8.3 (Building a utility disk) 這節。保留一份相同版本之備份用工具程式的copy是比較好的，這個備份用工具程式被用來製作備份，如此你就不用浪費時間在嘗試安裝不能讀取你備份磁帶的版本。

請確定你包括了以下程式： `init`, `getty` 或相等的程式, `login`, `mount`, 某種可以執行你rc scripts 的 shell, 以及一個從 `sh` 指向這個 shell 的 link。

4.3.4 /lib

在 `/lib` 中，你要放入必要的共享函式庫 (libraries) 與載入程式 (loaders)。如果無法在你的 `/lib` 目錄中找到必要的函式庫，那麼系統將不能夠開機。如果你很幸運，你可能會看到告訴你爲什麼會發生如此情況的錯誤訊息。

近來每一個程式至少都要求 `libc` 函式庫， `libc.so.`N，其中 N 是目前版本的編號。請檢查你的 `/lib` 目錄。 `Libc.so.N` 通常是一個 symlink，它指向一個具有完整版本編號的檔名：

```
% ls -l /lib/libc*
-rwxr-xr-x   1 root     root      4016683 Apr 16 18:48 libc-2.1.1.so*
lrwxrwxrwx   1 root     root           13 Apr 10 12:25 libc.so.6 -> libc-2.1.1.so*
```

在這個情況下，你會想要 `libc-2.1.1.so`。爲了找到其它函式庫，你應該要看過所有你打算包括的二進位檔，並且用 `ldd` 指令檢查它們的相依性。舉例來說：

```
% ldd /sbin/mke2fs
libext2fs.so.2 => /lib/libext2fs.so.2 (0x40014000)
libcom_err.so.2 => /lib/libcom_err.so.2 (0x40026000)
libuuid.so.1 => /lib/libuuid.so.1 (0x40028000)
libc.so.6 => /lib/libc.so.6 (0x4002c000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

在右手邊的每一個檔案是一定要有的。有些檔案可能是一個 symbolic link。

請注意某些函式庫 相當地大，而且並不能輕易地配合你的 root filesystem。舉例來說，上述的 `libc.so` 大約有 4 megabytes。因此，在你 copy 它們到你 root filesystem 的同時，你可能需要除去一些函式庫。請看 8.1 (Reducing root filesystem size) 這一節以了解 instructions。

在 `/lib` 內，你也必須包括一個 loader 供這些函式庫使用。這個 loader 不是 `ld.so` (給 A.OUT 函式庫使用)，就是 `ld-linux.so` (給 ELF 函式庫使用)。較新版的 `ldd` 會如同上述的例子，正確地告訴你需要哪一種 loader，然而舊版的就不會。如果你並不確定你需要哪一種 loader，就對函式庫執行 `file` 指令。舉例來說：

```
% file /lib/libc.so.4.7.2 /lib/libc.so.5.4.33 /lib/libc-2.1.1.so
/lib/libc.so.4.7.2: Linux/i386 demand-paged executable (QMAGIC), stripped
/lib/libc.so.5.4.33: ELF 32-bit LSB shared object, Intel 80386, version 1, stripped
/lib/libc-2.1.1.so: ELF 32-bit LSB shared object, Intel 80386, version 1, not stripped
```

`QMAGIC` 指出 `4.7.2` 版是給 A.OUT 函式庫使用，而 `ELF` 指出 `5.4.33` 以及 `2.1.1` 版是給 ELF 使用。

請 copy 你所需要的指定 loader(s) 到你所建立的 root filesystem。針對所包括的二進位檔，函式庫與 loaders 應該被 仔細地檢查 。如果 kernel 不能載入所需的函式庫，那麼 kernel 就會在沒有錯誤訊息的情況下停止運作。

4.4 對 PAM 與 NSS 的提供 – Providing for PAM and NSS

你的系統可能會需要動態地載入未被 `ldd` 所見的函式庫。如果你沒有提供函式庫給這些項目，那麼你會在登錄 (log in) 或使用你的 bootdisk 時遭遇到困難。

4.4.1 PAM (Pluggable Authentication Modules)

如果你的系統使用 PAM (Pluggable Authentication Modules)，那麼你必須在你的 bootdisk上為 PAM 做一些預備。簡而言之， PAM 是一種複雜精密的模組化方法，針對使用者認證以及控制使用者對服務的存取。一個決定你的系統是否使用 PAM 的簡單方法，是對你的 `login` 可執行檔執行 `ldd`；如果輸出包括 `libpam.so`，你就需要 PAM。

幸運地，安全性通常並非 bootdisk 所關心的議題，因為任何對機器有實際存取權的人，通常能做任何他們無論如何想做的事。因此，你可以有效地關閉 PAM，只要在你的 root filesystem 造出一個簡單的 `/etc/pam.conf` 檔，這個檔看起來像這樣：

```
OTHER   auth       optional    /lib/security/pam_permit.so
OTHER   account    optional    /lib/security/pam_permit.so
OTHER   password   optional    /lib/security/pam_permit.so
OTHER   session    optional    /lib/security/pam_permit.so
```

請記得也 copy `/lib/security/pam\_permit.so` 這個檔到你的 root filesystem。這個函式庫只有大約 8K，因此它只佔用極小量的 overhead。

請記得這個組態設定允許任何人對這台機器上的檔案以及服務有完整的存取權。如果你因某種理由而在乎你 bootdisk 的安全性，那麼你就必須 copy 一些或是全部你硬碟的 PAM setup 到你的 root filesystem。請確定曾仔細地讀過 PAM 文件，並且 copy 任何在 `/lib/security` 中所需要的函式庫到你的 root filesystem 上。

你同時必須包含 `/lib/libpam.so` 到你的 bootdisk 上。但是你已知這件事，因為你已對 `/bin/login` 執行過 ldd，這動作顯示了其相依性。

4.4.2 NSS (Name Service Switch)

如果你正使用 glibc (aka libc6)，你就必須為名稱服務 (name switch) 進行預備，否則你會無法 log in。`/etc/nsswitch.conf` 這個檔案控制資料庫對各式服務之搜尋 (lookups)。如果你並不打算從網路上存取服務 (比如說： DNS或 NIS 搜尋)，那麼你只需要準備一個簡單的 `nsswitch.conf` 檔，這個檔案看起來像這樣：

```
passwd:     files
shadow:     files
group:      files
hosts:      files
services:   files
networks:   files
protocols:  files
rpc:        files
ethers:     files
```

```
netmasks:   files
bootparams: files
automount:  files
aliases:    files
netgroup:   files
publickey:  files
```

這樣就指定每一項服務只被 local files 提供。你同時需要包括 /lib/libnss_files.so.X，在此 X 是 1 的話是指 glibc 2.0，而 2 的話是指 glibc 2.1。這個函式庫將被以動態方式載入以處理檔案搜尋。

如果你打算從你的 bootdisk 存取網路，那麼你會想要製作一個更精巧複雜的 nsswitch.conf 檔。細節請參考 nsswitch man page。最後，請記得你必須為你所指定的每一項 服務 *(service)*，把 /lib/libnss_*service*.so.1 檔放入 bootdisk 中。

4.5 模組 – Modules

如果你有一個模組化的 kernel，你必須考量在開機後，你想要從你的 bootdisk 上載入哪一種模組。你可能會想要包括 ftape 與 zftape 模組 (如果你的備份磁帶是floppy tape)，以及給 SCSI 設備用的模組 (如果你有 SCSI 設備)，也可能是 PPP 或 SLIP 支援的模組 (如果你在緊急情況下想要存取網路)。

這些模組可能會被放在 /lib/modules。你同時也應該包括 insmod, rmmod 與 lsmod。根據你是否想要自動地載入模組，你可能也要包括 modprobe, depmod 與 swapout。如果你使用 kerneld，請與 /etc/conf.modules 一起包括進來。

然而，使用模組的主要優點在於你可以把非關鍵 (non-critical)模組移到 utility disk 上，在需要用到時才載入，這樣在你的 root disk 上會使用比較少的空間。如果你要處理許多不同的設備，這個方法比建立一個內建支援許多設備的巨大 kernel 來得好。

請注意，為了啓動 (boot) 一個壓縮的 ext2 filesystem，你必須有 ramdisk 與內建 ext2 支援。 它們不能夠以模組的方式被提供。

4.6 一些最後的細節 – Some final details

某些系統程式，諸如 login，當 /var/run/utmp 檔與 /var/log 目錄不存在時，會發出警告。所以：

```
mkdir -p /mnt/var/{log,run{
touch /mnt/var/run/utmp
```

最後，在你設定 (set up) 完所有你所需的函式庫後，執行 ldconfig 以在 root filesystem上重新製作 (remake) /etc/ld.so.cache。這個 cache 會告訴 loader 到哪裡找到函式庫。要重新製作 ld.so.cache，請下達以下指令：

```
chdir /mnt; chroot /mnt /sbin/ldconfig
```

chroot 是必要的，因為 ldconfig 總是會為 root filesystem 重新製作 cache。

4.7 Wrapping it up

一旦你完成 root filesystem 的建構工作，就 unmount 它，將之 copy 成一個檔案並壓縮它：

```
umount /mnt
dd if=DEVICE bs=1k | gzip -v9 > rootfs.gz
```

結束後，你會有一個名為 `rootfs.gz` 的檔案，這就是你被壓縮過的 root filesystem。你應該檢查它的 size 以確保它能放在一張軟碟片上；如果不行，你就必須回溯並移除一些檔案。 8.1 (Reducing root filesystem size) 這節有一些提示，告訴你有關減少 root filesystem 的 size。

5 選擇一個 kernel – Choosing a kernel

現在，你已有一個完整的被壓縮過的root filesystem。下一步是要建立或是選擇一個 kernel。在大部分的情況下， copy 你目前所使用的 kernel 並且從它啟動軟碟片是可能的。然而，會有一些情況，你會希望建立一個獨立的 (separate) kernel。

理由之一是 size 的考量。如果你正建立一張單一的 boot/root 磁片， kernel 將會是磁片上最大的檔案之一，也因此你必須盡可能地減少 kernel 的 size。為了減少 kernel size，請用能支援所想要的系統之必要最起碼的設備來建立 kernel。這是指丟去你所不想要的每一項。網路功能是可以丟去的好選擇，另外還有當運作你的 boot/root filesystem 時，任何你所不需要的磁碟機與其它設備的支援。如同前述，你的 kernel 必須有 ramdisk 與 ext2 支援內建於其中。

如果你已完成一套最起碼的 (minimum) 設備組合以便於將之放入 kernel 中，那麼接下來你需要開始進行要把什麼加入回來。或許一套 boot/root 磁片組之最常見用途，是為了要能檢查與回復 (restore) 一個已損毀的 root file system，而為了達成這個目標，你需要 kernel 的支援。舉例來說，如果你的備份都放在磁帶上，並使用 Ftape 存取你的磁帶機，那麼，如果你失去了你目前的root drive與含有 Ftape 的 drive，你將不能夠從你的備份磁帶進行回復儲存。你將必須重新安裝 Linux，下載並重裝 ftape，然後試著讀取你的備份。

此處的要點是，無論你已加入什麼 I/O 支援到你的 kernel 內以支援備份，你同時也應該把它們加入到你的 boot/root kernel 中。

實際建造 kernel 的程序詳述在隨附於 kernel 的文件中。你可以輕易地照著指示進行，所以可以 `/usr/src/linux` 為起點。如果你在建造 kernel 上有困難，你或許不應該企圖不擇手段地建造 boot/root systems。請記得用 “`make zImage`”壓縮 kernel。

6 把它們放在一起：製作磁片(組) – Putting them together: Making the diskette(s)

進行到這裡，你已經有一個 kernel 與一個壓縮的 root filesystem。如果你正在製作一張 boot/root 磁片，請檢查它們的 size，以確定它們都能放在同一張磁片上。如果你正在製作一套兩張磁片的 boot + root 磁片組。請檢查 root filesystem 以確定它能放在一張磁片上。

你應該決定是否使用 LILO 以啟動 bootdisk的 kernel。替代的方法是直接把 kernel copy 到磁片上，然後不使用 LILO 開機。使用 LILO 的好處在於讓你能夠提供一些參數給 kernel，這些參數對初始化硬體來說可能是必要的(請檢查你系統上的 `/etc/lilo.conf` 檔。如果這個檔存在，而且有一行像“`append=...`”，你可能需要這個特點 (feature))。使用 LILO 的缺點在於建造 bootdisk 變得更為複雜，並且佔用稍嫌

較多的空間。你將必須設定一個小型而獨立的 filesystem，我們稱之爲 *kernel filesystem*，在其中你傳送 kernel 以及一些 LILO 所需的其它檔案。

如果你將要使用 LILO，請繼續讀下去；如果你要直接傳送 kernel，先跳過此，直接到 6.2 (Without using LILO) 這一節。

6.1 用 LILO 傳送 kernel – Transferring the kernel with LILO

你所必須做的第一件事是爲 LILO 編寫一個小型的組態檔。這個檔看起來像這樣：

```
boot      =/dev/fd0
install   =/boot/boot.b
map       =/boot/map
read-write
backup    =/dev/null
compact
image     =KERNEL
label     =Bootdisk
root      =/dev/fd0
```

關於這些參數的說明請看 LILO's user documentation。你或許也會想從你硬碟上的 `/etc/lilo.conf` 檔，加入一行 append=... 到這個組態檔中。

把這個組態檔存成 `bdlilo.conf`。

現在你必須製作一個小型的檔案系統，我們稱之爲 *kernel filesystem*，有別於 root filesystem。

首先，指出這個 filesystem 應該會有多大。用 blocks 計算你 kernel 的 size (這個 size 用 "`ls -l KERNEL`" 顯示出來，是被 1024 所除並強迫進位)，然後加 50。這個 50 blocks 是估計的空間，爲 inodes 與其它檔案所需。如果你想要，你可以精確地計算這個數字，或是就用 50 就好了。如果你正在製作兩張磁片的磁片組，你會高估這個空間，因爲無論如何第一張磁片只給 kernel 使用。我們稱這個數字會 `KERNEL_BLOCKS`。

請把軟碟片放入軟碟機 (爲求簡便，我們假設是 `/dev/fd0`)，然後在其上製作 ext2 kernel filesystem。

```
mke2fs -i 8192 -m 0 /dev/fd0 KERNEL_BLOCKS
```

"`-i 8192`" 指定我們想要每 8192 bytes 一個 node。接下來， mount 這個 filesystem，移除 `lost+found` 目錄，然後爲 LILO 造出 `dev` 與 `boot` 這兩個目錄：

```
mount /dev/fd0 /mnt
rm -rf /mnt/lost+found
mkdir /mnt/{boot,dev}
```

再來，造出設備檔 `/dev/null` 與 `/dev/fd0`。不用尋找設備號碼，你只要從你的硬碟上用 -R copy它們：

```
cp -R /dev/{null,fd0} /mnt/dev
```

LILO 需要一份它的 boot loader 的 copy， `boot.b`，你可以從你的硬碟得到它。它通常是放在 `/boot` 目錄內。

```
cp /boot/boot.b /mnt/boot
```

最後，隨附你的 kernel， copy 你在上一節所編寫的 LILO 組態檔。它們兩者可以被放在根目錄下。

```
cp bdlilo.conf KERNEL /mnt
```

LILO 所需的每一項現在都在 kernel filesystem上，所以你已準備好去執行它。 LILO 的 `-r` 旗號(flag)被用作安裝 boot loader 在某個其它的 root 上：

```
lilo -v -C bdlilo.conf -r /mnt
```

LILO 應該可以在無錯的情況下執行，在此之後，你可以看看你的 kernel filesystem，應該長得像這樣：

```
total 361
  1 -rw-r--r--   1 root     root          176 Jan 10 07:22 bdlilo.conf
  1 drwxr-xr-x   2 root     root         1024 Jan 10 07:23 boot/
  1 drwxr-xr-x   2 root     root         1024 Jan 10 07:22 dev/
358 -rw-r--r--   1 root     root       362707 Jan 10 07:23 vmlinuz
boot:
total 8
  4 -rw-r--r--   1 root     root         3708 Jan 10 07:22 boot.b
  4 -rw-------   1 root     root         3584 Jan 10 07:23 map
dev:
total 0
  0 brw-r-----   1 root     root       2,   0 Jan 10 07:22 fd0
  0 crw-r--r--   1 root     root       1,   3 Jan 10 07:22 null
```

如果檔案 size 和你自己的 filesystem 有稍微不同，請不用擔心。

現在請把磁片留在軟碟機內，然後前進到 6.3 (Setting the ramdisk word) 這一節。

6.2 不使用 LILO 來傳送 kernel – Transferring the kernel without LILO

如果你 不 使用LILO，那麼就用 `dd` 指令來傳送 kernel 給 bootdisk：

```
% dd if=KERNEL of=/dev/fd0 bs=1k
353+1 records in
353+1 records out
```

在這個例子中， `dd` 寫入了 353 個完整記錄(records) + 1個partial record，所以 kernel 佔用了 354 個軟碟片的 blocks。這個數字稱爲 `KERNEL_BLOCKS`，請記得它，這個數字要給下一節使用。

最後，請設定 root device 爲軟碟片自己，然後再設定 root 要被載入成 read/write：

```
rdev /dev/fd0 /dev/fd0
rdev -R /dev/fd0 0
```

在第二個 `rdev` 指令中，請小心地使用 `-R`。

6.3 設定ramdisk – Setting the ramdisk word

在 kernel image內的是 *ramdisk word*，它伴隨其它選項，指定了 root filesystem 應該在哪裡被找到。這個 word 會被以 `rdev` 指令，來進行存取與設定，而它的內容被解釋如下：

```
bits  0-10:     Offset to start of ramdisk, in 1024 byte blocks
bits 11-13:     unused
bit     14:     Flag indicating that ramdisk is to be loaded
bit     15:     Flag indicating to prompt before loading rootfs
```

如果 bit 15 被設定，在開機時 (boot-up)，會提示你把一張新的軟碟片放入軟碟機中。這對一套雙磁片的開機磁片組來說是必要的。

依據你是建造一張單一的 boot/root 磁片，還是兩張 "boot + root" 的磁片組，這裡會產生兩種情況。

1. 如果你是建造一張單一磁片，壓縮的 root filesystem 會被放置在 kernel 之後，所以 offset 會是第一塊 free block (這個值應該會與 `KERNEL_BLOCKS` 相等)。 Bit 14 會被設定為 1，而 bit 15 會被設定為 0。

   舉例來說，假設你正建造一張單一磁片，而 root filesystem 將自 block 253 (10進位)開始。那麼，ramdisk word 值應該是 253 (10進位)， bit 14 設為 1 而 bit 15 設為 0。要計算這個值，你可以簡單地加一加這個 10 進位數字。 253 + (2^14) = 253 + 16384 = 16637。如果你並不十分了解這個數字從何而來，把這數字放到工程計算機上，然後把它轉換為 2 進位，

2. 如果你建造的是一套雙磁片組，那麼 root filesystem 將自第二張磁片的 block 0 開始，所以 offset 為 0。 Bit 14 設為 1 而 bit 15 設為 1。在這個情況下， 10 進位值是 2^14 + 2^15 = 49152。

在仔細地為 ramdisk word 計算後，請用 `rdev -r` 設定它。請注意要使用 *10* 進位 值。如果你使用 LILO，傳給 `rdev` 的引數在此應該是 *mounted kernel path*，比如說 `/mnt/vmlinuz`；如果你用 `dd` 來 copy kernel，替換成使用軟碟機名稱(比如說 `/dev/fd0`)。

```
rdev -r KERNEL_OR_FLOPPY_DRIVE  VALUE
```

如果你使用 LILO，現在請把磁片 unmount。

6.4 傳送root filesystem – Transferring the root filesystem

最後一步是要傳送 root filesystem。

- 如果 root filesystem 將被放置在做為 kernel 之 相同 磁片上，請使用 `dd` 指令及其 `seek` 選項傳送它，這會指定要跳過多少個 blocks：

  ```
  dd if=rootfs.gz of=/dev/fd0 bs=1k seek=KERNEL_BLOCKS
  ```

- 如果 root filesystem 會被放置在 第二張 磁片上，請自磁碟機中拿走第一張軟碟片然後放入第二張磁碟片，接著將 root filesystem 傳送到其上：

  ```
  dd if=rootfs.gz of=/dev/fd0 bs=1k
  ```

恭喜你，已經完成了！

在把 bootdisk 挪為緊急使用之前，請記得先測試它 !如果你的成品不能執行，請繼續讀下去。

7 問題解決 – Troubleshooting, or The Agony of Defeat

當建造 bootdisks 時，開始的幾次嘗試結果常常是無法開機。建造一張 root disk 的一般方法是組合來自你現有系統的元件 (compnents)，接著嘗試與得到這個以磁片爲基礎的系統 (diskette-based system)，直到它在 console 上顯示訊息。一旦它開始與你交談，這場戰鬥就結束一半了，因爲你可以了解它在抱怨什麼，而且你可以解決個別的問題直至系統能平順地運作。如果系統停止運作而沒有任何說明，要找到這原因會是一件困難的事。爲了讓系統能啓動到能與你交談的階段，這需要使用到好幾種元件，而且這些元件要經過正確地組態。以下是調查系統無法與你交談之問題的建議程序：

- 你會看到一行訊息像這樣：

  ```
  kernel panic: VFS: Unable to mount root fs on XX:YY
  ```

  這是常見的問題，而它只有一些原因。首先，請根據 device code 清單檢查 device *XX:YY*；它是正確的 root device 嗎 ?如果不是，那你可能沒有做過 `rdev -R`，或者是你是對錯誤的 image 執行 rdev -R。如果 device code 是正確的，那麼請仔細地檢查被編譯到你 kernel 內的 device 驅動程式。請確定它有內建軟碟片、 ext2 filesystem 支援。

- 如果你看到許多錯誤訊息像是：

  ```
  end_request: I/O error, dev 01:00 (ramdisk), sector NNN
  ```

  這是 ramdisk driver 所報告的 I/O 錯誤訊息，可能是因爲 kernel 正嘗試寫入超越了設備的結尾。你的 ramdisk 因爲太小而不能持有你的 root filesystem。請檢查你的 bootdisk kernel 的初始化訊息，找一行像是：

  ```
      Ramdisk driver initialized : 16 ramdisks of 4096K size
  ```

  請針對 root filesystem未經壓縮的 size檢查上述的 size。如果 ramdisks 不夠大，那就讓它更大吧。

- 請檢查 root disk 實際上有包含你所認爲應該被包含的目錄。 Copy 到錯誤的 level 是容易犯的錯誤，以致於到最後在你的 root 磁碟片上，你會有像是 `/rootdisk/bin` 的目錄，而非 `/bin`。

- 請檢查是否有 `/lib/libc.so`，而它是否有與你硬碟內 `/lib` 目錄中所顯示之相同 link。

- 請檢查你既存系統 `/dev` 目錄內的 symbolic links 是否同時也存在於你的 root 磁碟片 filesystem 上，在此，那些 links 是連往你已包括在你 root 磁碟片上的 devices。尤其是在很多情況下，`/dev/console` links是最基本的。

- 請檢查你是否已包括了 `/dev/tty1`,`/dev/null`,`/dev/zero`,`/dev/mem`,`/dev/ram` 與 `/dev/kmem` 這些檔案。

- 請檢查你的 kernel 組態設定 - 對於直到登錄點 (login point) 所需之所有資源的支援必須被內建，不能使用模組。所以 *ramdisk* 與 *ext2* 支援必須被內建 。

- 請檢查你的 kernel root device 與 ramdisk 設定是否正確。

以上已經涵蓋一般部分，以下是一些更特定的檔案與檢查項目：

1. 請確定 `init` 是以 `/sbin/init` 或 `/bin/init` 的形式被包括進來。並且確認它是可執行的。

2. 請執行 `ldd init` 以檢查 init 的函式庫。通常這個就是 `libc.so`，但是請無論如何還是檢查一下。請確定你已放入了必備的函式庫與 loaders。

3. 請確定你把正確的 loader 給你的函式庫 – `ld.so` 是給 a.out 而 `ld-linux.so` 是給 ELF。

4. 請針對 `getty` (或某種類似 `getty` 的程式，諸如 `agetty`, `mgetty` 或 `getty_ps`)的呼叫，檢查你 bootdisk filesystem上的 `/etc/inittab`。請針對你硬碟的 `inittab` 檢查兩次。再檢查你所使用的程式的 man pages 以確定這些是有意義的。 `inittab` 可能是最詭異的部分，因爲它的語法與內容，依據所使用的 init 程式與系統本質而有所不同。解決的唯一之道就是去閱讀 `init` 與 `inittab` 的 man pages，然後再正確地做出既存系統開機時所做的事。請檢查以確定 `/etc/inittab` 有一個系統初始化的記錄 (entry)。它應該包括一個指令以執行必定存在的系統初始化 script。

5. 如同 `init`，對你的 `getty` 執行 `ldd` 以了解它需要什麼，同時確定必備的函式庫檔案與 loaders 是否被放入你的 root filesystem。

6. 請確認你已放入一個 shell 程式 (e.g., `bash` or `ash`)，它能夠執行你所有的 rc scripts。

7. 如果在你的救援磁片上，有一個 `/etc/ld.so.cache` 的檔案，請重新製作 (remake) 它。

如果 `init` 啓動，但是你卻得到一行訊息像是：

```
Id xxx respawning too fast: disabled for 5 minutes
```

它是來自於 `init`，通常指出 `getty` 或 `login` 被終結只要 init 啓動。請檢查 `getty` 與 `login` 的可執行檔與相依的函式庫。請確定在 `/etc/inittab` 內的呼叫 (invocations) 是正確的。如果你得到來自 `getty` 的奇怪訊息，它可能是指 `/etc/inittab` 內的呼叫格式是錯誤的。

如果你得到一個 login 提示 (prompt)，然後你輸入一個有效的 login name，但是系統卻立即提示你要輸入另一個 login name，那麼這個問題可能是出在 PAM 或 NSS。請看 4.4 (PAM and NSS) 這一節。問題也可能是你使用 shadow passwords 而你卻沒有 copy `/etc/shadow` 到你的 bootdisk 上。

如果你嘗試去執行某個可執行檔，諸如 `df`，而這個執行檔是在你的救援磁片上，執行時卻出現像這樣的訊息： `df: not found`，這時請檢查兩件事：(1)請確定包含二進位檔的目錄有設定在你的 PATH，(2)請確定你有程式所需的函式庫 (與 loaders)。

8 其它各種主題 – Miscellaneous topics

8.1 減少 root filesystem 的 size – Reducing root filesystem size

有時 root filesystem 會因太大而無法塞入一張軟碟片內，甚至在壓縮過後還是這樣。這裡有一些方法可減少 filesystem 的 size：

1. 增加磁碟片密度 *(density)*。 依照預設，軟碟片會以 1440 K 來進行格式化，但是現在也有更高密度的格式。 `fdformat`會爲以下 sizes 的磁片進行格式化： 1600, 1680, 1722, 1743, 1760, 1840, 以及 1920。大部分的 1440 K 軟碟機可支援 1722 K，我一直都是使用這個來製作 bootdisks。請參考 `fdformat` man page與 `/usr/src/linux/Documentation/devices.txt`。

2. 替換你的 *shell*。 一些在 Linux 上廣受歡迎的 shells，諸如 `bash` 與 `tcsh`，是相當大且需要許多函式庫。 Light-weight 的選擇於是存在，諸如 `ash`, `lsh`, `kiss` 與 `smash`，它們小很多而且只需要很少 (或是不需要)函式庫。大部分這些用來代替的 shells 可以在 `<http://metalab.unc.edu/pub/`

Linux/system/shells/> 找得到。請確定你所使用的任何 shell，能夠執行你放在你 bootdisk 內的所有 rc 檔案中的指令。

3. 刪去函式庫與二進位檔。 很多函式庫與二進位檔一般而言是未被刪去的 (包括debugging symbols)。如果如此，對這些檔案執行 file 會告訴你 "not stripped"。 當 copy 二進位檔到你的 root filesystem 時，這是一個好練習去使用：

   ```
   objcopy --strip-all FROM TO
   ```

   當 copy 函式庫時，請確定使用的是 strip-debug 而不是 strip-all。

4. 如果你在製作 root filesystem 時，刪除或移動過多檔案，請再重新製作一次。請看上述關於不要在 filesystem 內有 dirty blocks 之重要性的注意事項。

5. 移動非關鍵檔案到工具磁片 *(utility disk)* 上 。 如果你的一些二進位檔對 boot 或 login 來說並非立即需要，那麼你就可以把它們移到工具磁片上。細節請看 8.3 (Building a utility disk) 這一節。你同樣可考慮把 modules 移到工具磁片上。

8.2 Non-ramdisk root filesystems

4 (Building a root filesystem) 這一節曾給予指示以建造一個壓縮的 root filesystem，它會在系統開機時被載入到 ramdisk。這個方法有許多優點，所以通常都採用此法。然而，一些只有一點點記憶體的系統無法負擔此法所需的 RAM，因此，這些系統必須使用直接從軟碟片掛上的 root filesystems。

這樣的 filesystems 實際上比壓縮的 root filesystems 更容易建造，因爲它們可以被造在一張磁片上，而非某個其它的設備上，另外，它們也不必被壓縮。當異於前述的指示時，我們將敘述該程序的要點。如果你選擇這樣做，請記得你可獲得的剩餘 空間會變少 。

1. 請計算你將有多少空間可供 root files 使用。

   如果你建造的是一張單一的 boot/root 磁片，你必須讓所有給 kernel 的 blocks 以及所有給 root filesystem 的 blocks 都能容納於同一張磁片上。

2. 使用 mke2fs，在適當 size 的軟碟片上造出一個 root filesystem。

3. 如同前述的方法殖入於此 filesystem。

4. 完成後， unmount 此 filesystem 並且傳送它使之成爲一個磁片檔案，但是 不要壓縮它 。

5. 以前述之方法，把 kernel 傳送到一張軟碟片上。當計算 ramdisk word 時， 把 *bit 14* 設成 *0*，以指示不要把 root filesystem 載入到 ramdisk。請以前述之方法執行 rdev 指令。

6. 如同以往，傳送此 root filesystem。

有好幾種捷徑你可採用。如果你建造的是一套雙磁片組，你可以直接把完整的 root filesystem建造在第二張磁片上，而且你並不需要把它傳送成一個硬碟檔案然後再存回磁片上。同樣地，如果你建造的是一張單一的 boot/root 磁片而且使用 LILO，你可以在整張磁片上建立 單一的 filesystem，包含 kernel、LILO files 與 root files，然後只要執行 LILO 做爲最後一步。

8.3 建造一張工具磁片 – Building a utility disk

建造一張工具磁片相對來說簡單多了 – 只要在一張已格式化的磁片上造出一個 filesystem，然後 copy 檔案於其上即可。爲了跟 bootdisk 一起使用它，請在系統啓動以後，用手動的方式 mount 它。

在前面的指示中，我們曾提及 utility disk 可以被 mount 做爲 `/usr`。在這個情況下，二進位檔可以被放在你 utility disk 之 `/bin` 目錄內，以便於將 `/usr/bin` 設入你的 path 中可以存取它們。二進位檔所需之其它函式庫被放在 utility disk 之 `/lib` 目錄內。

當設計 utilitu disk 時，有幾個重點要記住：

1. 不要把關鍵的系統二進位檔或函式庫放到 utility disk 上，因爲直到系統順利啓動之前， utility disk 不會被掛上，其上的檔案也無法供系統使用。
2. 你不能同時存取一張軟碟片與一台磁帶機。意指如果你有一台磁帶機，那麼當你正在使用 (mount) utility disk 時，你將不能存取那台磁帶機。
3. 存取 utility disk 上的檔案會很慢。

D (Sample utility disk directory listing) 提供了utility disk的檔案之樣本。以下是一些建議關於一些你會發現有用的檔案：檢查與操作磁片(`format`, `fdisk`)、filesystems (`mke2fs`, `fsck`, `debugfs`, `isofs.o`)的程式，小型的文書編輯器 (`elvis`, `jove`)，壓縮與檔案工具 (`gzip`, `tar`, `cpio`, `afio`), 磁帶工具 (`mt`, `tob`, `taper`)，通訊工具 (`ppp.o`, `slip.o`, `minicom`)與給設備使用的工具 (`setserial`, `mknod`)。

9 How the pros do it

您可能已經注意到，那些被諸如 Slackware、RedHat、Debian 等主要 distribution 所使用的 bootdisks，似乎比本文中所描述的還要來得複雜許多。專業的 distribution bootdisks 以在此所提出的相同原則爲基礎，但是採用各式各樣的技巧，因爲它們的 bootdisks 有一些額外的需求。第一，它們必須能夠在各種不同的硬體上運作，所以它們必須能夠與使用者互動，並且能載入各式各樣的設備驅動程式。第二，它們必須準備以許多不同的安裝選項、不同的自動化程度來運作。最後， distribution bootdisks 通常結合了安裝磁片與救援磁片的能力。

某些 bootdisks 使用名爲 *initrd*(*initial ramdisk*)的特性。這個特性大約在 2.0.x 版時引入，它允許 kernel 以兩階段開機。當 kernel 開機時，它從 boot disk 載入一個initial ramdisk映像檔。這個 initial ramdisk 是一個 root filesystem，包含一個在眞正的 root fs 被載入之前所執行的程式。這個程式通常會檢查作業環境，以及/或 要求使用者選擇不同的開機選項，例如選擇從哪一個設備載入眞正的 rootdisk。一般來說，它會載入未被內建在 kernel 內的額外模組。當這個初始化 (initial) 程式結束 (exit) 時， kernel 就載入眞正的 root 映像檔，開機動作將依一般情況繼續執行下去。若要進一步得知 `initrd` 的資訊，請詳閱你機器上的 */usr/src/linux/Documentation/initrd.txt* `<file:/usr/src/linux/Documentation/initrd.txt>`，以及 `<ftp://elserv.ffm.fgan.de/pub/linux/loadlin-1.6/initrd-example.tgz>`。

以下是每一家 distribution的安裝磁片如何運作的摘要，這份摘要以檢閱 (inspect) 它們的filesystems 以及/或 它們的原始碼爲基礎。我們不保證這份資訊完全正確無誤，或是自從版本noted以來，這些運作方式未被改變。

Slackware(v.3.1) 使用一種與 6.1 (Transferring the kernel with LILO) 這節所描述類似的前向式 (straight-forward) LILO 開機法。 Slackware 的 bookdisk 利用 LILO 的 訊息 參數印出一個 bootup 訊息 (“`Welcome to the Slackware Linux boot kernel disk!`”)。假如必要的話，這裡會指示使用者輸入一個開機參數行(boot parameter line)。在開機之後，一個root檔案系統會從第二張磁片載入。此時使用者喚起

(invoke)一個 `setup` script，這個script將啓動安裝程序。Slackware並非使用一個模組化 kernel，相反地，它提供許多不同的 kernel，並且靠使用者自己選擇一個符合 他/她 硬體需求的 kernel。

RedHat(v.4.0) 也使用 LILO 開機法。它從第一張磁片載入一個壓縮的 ramdisk，這動作執行一個 custom `init` 程式。這個程式查詢驅動程式，然後，假如必要的話，從 supplemental 磁片載入額外的檔案。

Debian(v1.3) 可能是最複雜的安裝磁片集。它使用 SYSLINUX loader 去安排各式各樣的載入選項，然後使用一個 `initrd` 映像檔指示使用者完成安裝程序。它顯然使用了一個客製化的 `init` 與一個客製化的 shell 兩者。

10 常見問題 (FAQ) 列表 – Frequently Asked Question (FAQ) list

Q. 我從我的 boot/root 磁片開機，但是什麼都跑不出來。我現在怎麼辦？

請看之前的 7 (Troubleshooting) 這節。

Q. Slackware/Debian/RedHat 的 bootdisk 如何運作？

請看之前的 9 (What the pros do) 這節。

Q. 我要如何以 XYZ 驅動程式製作一張開機磁片？

最簡單的方法是去從離你最近的 Slackware 映射站拿到一個 Slackware 的 kernel。 Slackware 的 kernel 是一般的 (generic) kernel，這些 kernel 企圖將許多設備的驅動程式儘可能地包含於其中，因此，假如你有一個 SCSI或IDE控制器，試試看，很有可能它的驅動程式會在 Slackware 的 kernel 當中。

找到 `a1` 目錄，並且依據你所擁有的控制器種類，選擇 IDE 或 SCSI 兩者之一的 kernel。對所選擇的 kernel 檢視其 xxxxkern.cfg 檔，並且去了解這份 kernel 中所擁有的驅動程式。如果你想要使用的設備在這份列表中，那麼這個符合的 kernel 就應該能用來開機。下載 xxxxkern.tgz 檔，並且用之前在有關 making boot disks 的章節中所描述之方法， copy 它到你的開機磁片中。

然後你必須檢查在 kernel 中的 root 設備，使用 `rdev zImage` 這個指令。接著 rdev 將顯示目前在 kernel 中的 root 設備。如果這和你所想要的 root 設備不同，那麼使用 `rdev` 改變它。舉例來說，我試的 kernel 設定到 `/dev/sda2`，但是我的 root SCSI partition 是 `/dev/sda8`。爲了能使用 root 磁片，你將必須使用這個指令 `rdev zImage` dev/fd0/。

假如你還想知道如何 set up 一張 Slackware 的 root 磁片，這就已經超出這份 HOWTO 的範圍，所以我建議你查閱 Linux Install Guide，或是設法取得 Slackware distribution。詳情請看在這份 HOWTO 中，標題爲 “Reference” 的章節。

Q. 我如何以新的檔案更新我的 root 磁片？

最簡單的方法是從 rootdisk copy 檔案系統回你所使用的那個 DEVICE (從之前的 4.2 (Creating the filesystem) 這節而得的)。然後掛上這個檔案系統並且進行改變動作。你必須記住你的 root 檔案系統從哪裡開始，以及它佔了多少 block：

```
dd if=/dev/fd0 bs=1k skip=ROOTBEGIN count=BLOCKS | gunzip > DEVICE
mount -t ext2 DEVICE /mnt
```

在完成改變之後，如同之前一樣進行下去 (在 4.7 (Wrapping it up) 這節中)，並且把 root filesystem 傳送回那張磁片上。如果你並沒有改變新的 root filesystem 的啓始位置，你應該不用再重傳 kernel，或是重新計算 ramdisk 的 word。

Q. 我要如何移除 LILO，好讓我能再用 DOS 開機？

這個問題並非眞的是 Bootdisk 的一個主題，只是這個問題常常被問到。在 Linux 的環境下，你可以執行：

```
/sbin/lilo -u
```

你也可以使用 dd 這個指令把被 LILO 所儲存的備份， copy 到開機磁區上。如果你想要這樣做，請參考 LILO 的文件。

在 DOS 與 Windows 的環境下，你可以使用這個 DOS 指令：

```
FDISK /MBR
```

MBR 是 Master Boot Record(主要開機記錄)的縮寫，它會用一個乾淨的 DOS 開機磁區，替換原本的開機磁區，而且這個動作不會影響 partition table。一些有潔癖的人 (purists) 並不同意這一點，可是就連 LILO 的原作者， Werner Almesberger，都同意此作法。這個作法簡單，而且有用。

Q. 如果我遺失了我的 *kernel* 與 我的開機磁片，我要如何開機 *?*

如果你並沒有一張已準備好的開機磁片，最簡單的作法可能是要依照你的磁碟控制器類型 (IDE 或 SCSI) 取得一份 Slackware kernel，如同之前所述的“我如何用XXX驅動程式製作一張開機磁片 ?”。然後你就可以用這個 kernel 開機，接著修理有損壞的地方。

你取得的 kernel 可能沒有與你想要的磁碟種類和 partition 相對應的 root 設備集。舉例來說， Slackware 的 generic SCSI kernel 有與 `/dev/sda2` 相對應的 root設備集，然而，我的 root Linux partition 是在 `/dev/sda8`。在此情況下， kernel 中的 root 設備將必須被改變。

你仍然可以改變 kernel 的 root 設備與 ramdisk 的設定，縱使你手上只有一個 kernel 和某種其它的作業系統，像 DOS。

`rdev` 藉著改變在 kernel 檔案裡，在固定位移位置 (offset) 上的值，來改變 kernel 的設定，因此，如果你手上有一個 hex 編輯器 (on whatever systems you do still have running)，你可以完成相同的事 – 例如在 DOS 環境下的 Norton Utilities Disk Editor。接著，你需要進行檢查，如果必要的話，你得改變 kernel 檔案裡，在以下位移位置上的值：

```
HEX(16進位)     DEC(10進位)  DESCRIPTION(用途描述)
0x01F8          504          RAMDISK word的低字元組 (Low byte of RAMDISK word)
0x01F9          505          RAMDISK word的高字元組 (High byte of RAMDISK word)
0x01FC          508          Root minor設備號碼 - 詳見其後
0X01FD          509          Root major設備號碼 - 詳見其後
```

關於 ramdisk word 之解釋，寫在之前的 6.3 (Setting the ramdisk word) 這節裡。

Major 與 minor 設備號碼必須設成你想要掛你的 root filesystem 於其上的設備。一些可供選擇的有用參考數值如下：

```
DEVICE          MAJOR MINOR
/dev/fd0            2     0    第一台軟碟機
/dev/hda1           3     1    在第一顆IDE硬碟上的partition 1
/dev/sda1           8     1    在第一顆SCSI硬碟上的partition 1
/dev/sda8           8     8    在第一顆SCSI硬碟上的partition 8
```

一旦你設定了這些值，接下來你可以將這個檔案寫入一張磁片內，你可以利用 Norton Utilities Disk Editor，不然就是利用名爲 `rawrite.exe` 的這支程式。這支程式在所有的 distribution 都找得到。這支

程式是一支在DOS環境下執行的程式，它能將檔案寫入一張 “raw” disk，從開機磁區開始寫入，而不是將檔案寫入檔案系統。如果你使用 Norton Utilities，你必須將檔案寫入一張實體磁片，從磁片的頭開始寫入。

Q. 我要如何製作 *boot/root* 磁片的額外備份 ?

因爲磁性媒介可能在一段時間後遭遇損害，你應該保留幾張你的救援磁片的備份，以防原來的那一片不能被電腦讀取。

製作任何磁片的備份，包括開機用與公用程式 (utility) 磁片，最簡單的方法是利用 dd 這個指令，去 copy 原來那張磁片的內容到你硬碟上的一個檔案之中，然後再用相同的指令將這個檔案 copy 回一張新磁片上。請注意，你並不需要，也不應該去 mount 這張磁片，因爲 dd利用raw設備介面。

要 copy 原磁片，請輸入以下指令：

```
dd if=DEVICENAME of=FILENAME
```

在此， DEVICENAME 是指這台磁碟機的設備名 (譯按：/dev/fd0)，而 FILENAME 是指(存在硬碟上的)輸出檔案檔名。省略 count 參數讓 dd 去copy整張磁片 (如果是高密度磁片，有 2880 個 blocks)。

要把從上面得到的檔案 copy 到一張新磁片上，請插入一張新磁片並輸入相反的指令：

```
dd if=FILENAME of=DEVICENAME
```

請注意，以上的討論是假設你只有一台軟碟機。假如相同種類的軟碟機你有兩台，你可以利用像下面的指令去 copy 磁片：

```
dd if=/dev/fd0 of=/dev/fd1
```

Q. 我如何能在每一次開機時，不用輸入 *“ahaxxxx=nn,nn,nn”* ?

當一台磁碟設備不能被自動偵測時，就有必要提供 kernel 指定的設備參數字串，諸如：

```
aha152x=0x340,11,3,1
```

這行參數字串可以利用 LILO 以數種方法提供給 kernel：

- 每次當系統以 LILO 開機時，你可以在命令列輸入它。但是每次這樣做會很煩。
- 你可以使用 LILO 的 lock 這個關鍵字，使 LILO 儲存這行命令列，並做爲預設的命令列，這樣每當開機時， LILO 就會使用相同的選項。
- 你可以在 LILO 的組態檔中，使用 append= 這個敘述。請注意參數字串必須被雙引號 (quotes) 夾住。

舉例來說，一行使用上述參數字串的樣本命令列將長成這樣：

```
zImage  aha152x=0x340,11,3,1 root=/dev/sda1 lock
```

如此將會傳遞設備參數字串給 kernel，同時也要求 kernel 把 root 設備設定到 /dev/sda1，並且儲存整行命令列，讓以後開機時都能再使用這個命令列。

以下是一個 APPEND 敘述的樣本：

```
APPEND = "aha152x=0x340,11,3,1"
```

請注意參數字串在命令列上 不能 被雙引號夾住，但是參數字串在 APPEND 敘述中 一定要 被雙引號夾住。

外也請注意，爲了能讓參數字串起作用， kernel 內必須有符合這個磁碟類型的驅動程式。如果沒有，那麼就沒有東西會去接受 (listen for) 這個參數字串，所以你將必須重新建造一個包含指定驅動程式的 kernel。有關重新建立 kernel 的細節，請至目錄 `/usr/src/linux` 閱讀其 README 檔，以及 Linux FAQ 與 Installation HOWTO。 Alternatively，你可以爲這種磁碟類型取得一個 generic kernel 並且安裝它。

在體驗 LILO 安裝之前，強烈建議讀者先讀過 LILO 文件。不小心使用 `BOOT` 敘述將會損壞 partition。

Q. 在開機的時候，出現了 *“`A: cannot execute B`”* 的錯誤訊息。爲什麼 ?

程式名稱有好幾種不同的寫法被寫死 (hardcoded) 在不同的公用程式裡。這些寫法不會在每個地方都發生，但是它們可能說明了爲什麼一個顯而易見的可執行檔，不能在你的系統上被發現，縱使你看到它就在那裡。你可以找出是否一個已知的程式有另一種被寫死的名稱，你可以利用 `strings` 這個指令，並且利用管道方法 (譯按： piping, |) 將輸出透過 `grep` 呈現出來。

已知關於寫死的例子有：

- 在某些版本中， `Shutdown` 有把 `/etc/reboot` 寫死在其程式碼內，所以 `reboot` 必須被放置在 `/etc` 這個目錄下。
- 因爲有不能夠找到 `init` 的 kernel 存在， `init` 已至少爲一個人帶來問題。

爲了修正這些問題，不是移動這些程式到正確的目錄，就是更改設定檔 (e.g. `inittab`)以指向正確的目錄。假如有所懷疑，就請先把程式如同它們在你硬碟上一樣，放入相同的目錄，並且如同它們顯現在你硬碟上一樣，使用相同的 `inittab` 與 `/etc/rc.d` 檔。

Q. 我的 *kernel* 支援 *ramdisk*，但是卻初始化 *0K* 的 *ramdisk*。爲什麼 ?

當 kernel 正在進行開機動作時，在問題發生之處，會出現一個像這樣的 kernel 訊息：

```
Ramdisk driver initialized : 16 ramdisks of 0K size
```

這可能是因爲 size 已被 kernel 參數在開機時設成 0。這可能是由於一個 overlooked LILO組態設定檔參數所導致：

```
ramdisk= 0
```

在某些較舊的 distributions 裡，這個會被包含在 LILO 組態設定檔樣本中，這放在那裡是爲了 override 任何 kernel 原先的設定。如果你有這樣的一行，請移除之。

請注意，如果企圖使用一個已被設定成 0 size 的 ramdisk，這樣的行爲將會導致不可預測的結果，同時也會讓 kernel 不知如何是好。

A 資源與指示 – Resources and pointers

當拿取一個套件時，除非你有好的理由，否則請一定要拿最新的版本。

A.1 預先做好的 Bootdisks – Pre-made Bootdisks

這些是 distribution bootdisks 的來源。 請選擇映射站台下載以減少這些主機的負荷。

- *Slackware bootdisks* <http://metalab.unc.edu/pub/Linux/distributions/slackware/bootdsks.144/>, *rootdisks* <http://metalab.unc.edu/pub/Linux/distributions/slackware/rootdsks/> 與 *Slackware* 映射站台 <http://www.slackware.com/getslack/>
- *RedHat bootdisks* <http://metalab.unc.edu/pub/Linux/distributions/redhat/current/i386/images/> 與 *Red Hat* 映射站台 <http://www.redhat.com/mirrors.html>
- *Debian bootdisks* <ftp://ftp.debian.org/pub/debian/dists/stable/main/disks-i386/current/> 與 *Debian* 映射站台 <ftp://ftp.debian.org/pub/debian/README.mirrors.html>

除了 distribution bootdisks 以外，也可以得到以下的救援磁片映像檔。除非有另外特別指定，否則這些都可在 <http://metalab.unc.edu/pub/Linux/system/recovery/!INDEX.html> 的目錄中找到。

- `tomsrtbt`, 由 Tom Oehser 製作，是一張以 kernel 2.0 爲基礎而製作出來的單片裝 boot/root 磁片，with a large set of features and support programs。它支援 IDE、 SCSI、磁帶、網路卡、 PCMCIA 等還有很多其它設備。其中有大約 100 種工具程式與工具可以用來修護與備份磁碟。此套件也包含一些 script 用來解譯與重建構映像檔，以便於在必要時可以加入新的 material。
- `rescue02`, 由 John Comyns 製作，是一張以 kernel 1.3.84 爲基礎而製作出來的救援磁片，其支援 IDE、 Adaptec 1542 與 NCR53C7,8xx。它使用 ELF 兩進位檔，但是卻有足夠的指令以利其能在任何系統上使用。它擁有能在開機後才被載入給所有其它 SCSI 卡使用的模組。但它也許不能在只有 4 mb 隨機存取記憶體的系統上執行，因爲它用到了 3 mb 的ram disk。
- `resque_disk-2.0.22`, 由 Sergei Viznyuk 製作，是一套以 kernel 2.0.22 爲基礎，內建支援 IDE 與許多不同的 SCSI 控制卡，以及 ELF/AOUT 的全功能 boot/root 磁片。同時也包含許多模組，以及用來修護及備份硬碟的有用工具程式。
- *`cramdisk`* <http://metalab.unc.edu/pub/Linux/system/recovery/images> 映像檔，以 kernel 2.0.23 爲基礎，可使用在 4 meg 與 8 meg 的機器上。它們包含了數學模擬器與網路工具 (PPP與 dialin script、NE2000、3C509)，或是平行埠 ZIP 磁碟機的支援。這些磁片映像檔可在備有 4MB 隨機存取記憶體的 386 主機上開機。 MSDOS 支援也被包含在其中，因此你可以從網路上下載它到 DOS partition 上。

A.2 救援套件 – Rescue packages

目前可以從 metalab.unc.edu 取得數種製作救援磁片的套件。利用這些套件，你可以指定包含一組檔案，接著軟體就會自動地進行 bootdisk 的製作 (自動化程度會有所不同)。請看 <http://metalab.unc.edu/pub/Linux/system/recovery/!INDEX.html> 以取得進一步的資訊。

請仔細檢查檔案日期。一些套件有數年未被更新，而這些套件將無法支援「載入至 ramdisk 之壓縮 root filesystem 」的製作。就目前所知， *Yard* <http://www.croftj.net/~fawcett/yard/index.html> 是唯一可支援此的套件。

A.3 LILO – the Linux loader

由 Werner Almesberger 撰寫。一個優秀的 boot loader，其文件包含了開機磁區內容的資訊，以及開機流程的初期階段。

從 <ftp://tsx-11.mit.edu/pub/linux/packages/lilo/> 以FTP下載。也可以從 Metalab 與映射站台內取得。

A.4 Linux FAQ 與 HOWTOs

這些文件可以從諸多來源中取得。這些文件可以從諸多來源中取得。請見 usenet 新聞論壇 `news.answers` 與 `comp.os.linux.announce`。

這些 FAQ 可以從 <http://linuxdoc.org/FAQ/Linux-FAQ.html> 中取得，而 HOWTOs 可以從 <http://linuxdoc.org/HOWTO/HOWTO-INDEX.html> 中取得。大部分 Linux 文件可以在 *The Linux Documentation Project homepage* <http://linuxdoc.org/> 中找得到。

A.5 Ramdisk使用方法 – Ramdisk usage

有關新的ramdisk 程式碼如何運作的完整敘述，可以在隨附於 Linux kernel 的文件中找到。請看 `/usr/src/linux/Documentation/ramdisk.txt`。這份文件是由 Paul Gortmaker 所編寫，同時包含了一節關於製作壓縮的 ramdisk。

A.6 Linux開機流程 – The Linux boot process

若想了解關於Linux開機流程的更多細節，以下有一些指示文件：

- 在 <http://linuxdoc.org/LDP/sag/c1596.html>*Linux System Administrators' Guide* 中，有一節是關於 booting。
- 在 <http://metalab.unc.edu/pub/Linux/system/boot/lilo/lilo-t-21.ps.gz>LILO "Technical overview" 中，擁有關於開機流程一直到 kernel 從何處被啓動之決定性 (definitive) 技術上 low-level 的敘述。
- Source code 是你永遠的指南。以下是一些與開機流程有關的 kernel files。如果你有 Linux 核心的原始碼，你可以在你機器上的 `/usr/src/linux` 之中找到這些檔案；此外， Shigo Yamguchi (*shigio@tamacom.com*)有非常棒的 *hypertext kernel browser* <http://www.tamacom.com/tour/linux/index.html>，可以用來讀 kernel source files。以下是一些可供參考的有關檔案：

  `arch/i386/boot/bootsect.S` and `setup.S` 包含 bootsector 自己的組合碼。 `arch/i386/boot/compressed/misc.c` 包含未壓縮的 kernel 程式碼。 `arch/i386/kernel/` 包含了 kernel 初始化程式碼的目錄。 `setup.c` 定義了 ramdisk 的 word。 `drivers/block/rd.c` 包含 ramdisk 的驅動程式。 `rd_load` 與 `rd_load_image` 這兩個程序從一個設備中載入區塊 (blocks) 到 ramdisk 內。 `identify_ramdisk_image` 這個程序決定找到的 filesystem 是什麼種類，還有它是否是被壓縮的 filesystem。

B LILO boot error codes

有關這些錯誤碼的問題時常在 Usenet 上被人提到，所以我們以服務的心情將它們表列如下。這份摘要引用自 Werner Almsberger 的 *LILO User Documentation* <http://metalab.unc.edu/pub/Linux/system/boot/lilo/lilo-u-21.ps.gz>。

當 LILO 載入自己時，螢幕上會顯示 LILO 這個字。每一個字母 (letter) 會在執行某個特定動作之前或之後被印在螢幕上。如果 LILO 在某點上不能完成任務，到此爲止所印出的字母可以用來辨認出發生了什麼問題。

(nothing)

LILO 完全沒有被載入。 LILO 不是沒有被安裝好，就是 LILO 開機磁區所在的 partition 並非使用中的 partition。

L

第一階段的 boot loader 已被載入並啓動，但是無法載入第二階段的 boot loader。這些二位數的錯誤碼指出問題屬於什麼種類。(請看 "Disk error codes" 這一節。)這個情況通常是指媒體錯誤或是幾何不相配 (geometry mismatch)(舉例來說，錯誤的磁碟參數)。

LI

第一階段的 boot loader 能夠載入第二階段的 boot loader，但是卻不能夠執行它。這個情形是因爲 geometry mismatch 抑或是在沒有執行 map installer 的情況下，移動了 `/boot/boot.b` 所造成。

LIL

第二階段的 boot loader 已被啓動，但是卻不能從 map file 載入 descriptor table。一般而言，這是因爲媒體錯誤或是 geometry mismatch 所造成。

LIL?

第二階段的 boot loader 被載入到一個不正確的位址。一般而言，這是因爲 subtle geometry mismatch，或是在沒有執行 map installer 的情況下，移動了 /boot/boot.b 所造成。

LIL-

Descriptor table 損毀。這個情形是因爲 geometry mismatch 抑或是在沒有執行 map installer 的情況下，移動了 /boot/map 所造成。

LILO

LILO 的所有部分均已成功載入。

如果在 LILO 正試著載入 boot image 時， BIOS 發出一個錯誤訊號，那麼相對映的 (respective)錯誤碼會顯示出來。錯誤碼的範圍從 `0x00` 到 `0xbb`。請看 LILO User Guide for an explanation of these。

C Root filesystem 列表樣本 – Sample root filesystem listings

```
/:
drwx--x--x   2 root     root         1024 Nov  1 15:39 bin
drwx--x--x   2 root     root         4096 Nov  1 15:39 dev
drwx--x--x   3 root     root         1024 Nov  1 15:39 etc
```

```
drwx--x--x   4 root     root         1024 Nov  1 15:39 lib
drwx--x--x   5 root     root         1024 Nov  1 15:39 mnt
drwx--x--x   2 root     root         1024 Nov  1 15:39 proc
drwx--x--x   2 root     root         1024 Nov  1 15:39 root
drwx--x--x   2 root     root         1024 Nov  1 15:39 sbin
drwx--x--x   2 root     root         1024 Nov  1 15:39 tmp
drwx--x--x   7 root     root         1024 Nov  1 15:39 usr
drwx--x--x   5 root     root         1024 Nov  1 15:39 var

/bin:
-rwx--x--x   1 root     root        62660 Nov  1 15:39 ash
-rwx--x--x   1 root     root         9032 Nov  1 15:39 cat
-rwx--x--x   1 root     root        10276 Nov  1 15:39 chmod
-rwx--x--x   1 root     root         9592 Nov  1 15:39 chown
-rwx--x--x   1 root     root        23124 Nov  1 15:39 cp
-rwx--x--x   1 root     root        23028 Nov  1 15:39 date
-rwx--x--x   1 root     root        14052 Nov  1 15:39 dd
-rwx--x--x   1 root     root        14144 Nov  1 15:39 df
-rwx--x--x   1 root     root        69444 Nov  1 15:39 egrep
-rwx--x--x   1 root     root          395 Nov  1 15:39 false
-rwx--x--x   1 root     root        69444 Nov  1 15:39 fgrep
-rwx--x--x   1 root     root        69444 Nov  1 15:39 grep
-rwx--x--x   3 root     root        45436 Nov  1 15:39 gunzip
-rwx--x--x   3 root     root        45436 Nov  1 15:39 gzip
-rwx--x--x   1 root     root         8008 Nov  1 15:39 hostname
-rwx--x--x   1 root     root        12736 Nov  1 15:39 ln
-rws--x--x   1 root     root        15284 Nov  1 15:39 login
-rwx--x--x   1 root     root        29308 Nov  1 15:39 ls
-rwx--x--x   1 root     root         8268 Nov  1 15:39 mkdir
-rwx--x--x   1 root     root         8920 Nov  1 15:39 mknod
-rwx--x--x   1 root     root        24836 Nov  1 15:39 more
-rws--x--x   1 root     root        37640 Nov  1 15:39 mount
-rwx--x--x   1 root     root        12240 Nov  1 15:39 mt
-rwx--x--x   1 root     root        12932 Nov  1 15:39 mv
-r-x--x--x   1 root     root        12324 Nov  1 15:39 ps
-rwx--x--x   1 root     root         5388 Nov  1 15:39 pwd
-rwx--x--x   1 root     root        10092 Nov  1 15:39 rm
lrwxrwxrwx   1 root     root            3 Nov  1 15:39 sh -> ash
-rwx--x--x   1 root     root        25296 Nov  1 15:39 stty
-rws--x--x   1 root     root        12648 Nov  1 15:39 su
-rwx--x--x   1 root     root         4444 Nov  1 15:39 sync
-rwx--x--x   1 root     root       110668 Nov  1 15:39 tar
-rwx--x--x   1 root     root        19712 Nov  1 15:39 touch
-rwx--x--x   1 root     root          395 Nov  1 15:39 true
-rws--x--x   1 root     root        19084 Nov  1 15:39 umount
-rwx--x--x   1 root     root         5368 Nov  1 15:39 uname
-rwx--x--x   3 root     root        45436 Nov  1 15:39 zcat

/dev:
lrwxrwxrwx   1 root     root            6 Nov  1 15:39 cdrom -> cdu31a
brw-rw-r--   1 root     root      15,   0 May  5  1998 cdu31a
```

```
crw-------   1 root     root       4,   0 Nov  1 15:29 console
crw-rw-rw-   1 root     uucp       5,  64 Sep  9 19:46 cua0
crw-rw-rw-   1 root     uucp       5,  65 May  5  1998 cua1
crw-rw-rw-   1 root     uucp       5,  66 May  5  1998 cua2
crw-rw-rw-   1 root     uucp       5,  67 May  5  1998 cua3
brw-rw----   1 root     floppy     2,   0 Aug  8 13:54 fd0
brw-rw----   1 root     floppy     2,  36 Aug  8 13:54 fd0CompaQ
brw-rw----   1 root     floppy     2,  84 Aug  8 13:55 fd0D1040
brw-rw----   1 root     floppy     2,  88 Aug  8 13:55 fd0D1120
brw-rw----   1 root     floppy     2,  12 Aug  8 13:54 fd0D360
brw-rw----   1 root     floppy     2,  16 Aug  8 13:54 fd0D720
brw-rw----   1 root     floppy     2, 120 Aug  8 13:55 fd0D800
brw-rw----   1 root     floppy     2,  32 Aug  8 13:54 fd0E2880
brw-rw----   1 root     floppy     2, 104 Aug  8 13:55 fd0E3200
brw-rw----   1 root     floppy     2, 108 Aug  8 13:55 fd0E3520
brw-rw----   1 root     floppy     2, 112 Aug  8 13:55 fd0E3840
brw-rw----   1 root     floppy     2,  28 Aug  8 13:54 fd0H1440
brw-rw----   1 root     floppy     2, 124 Aug  8 13:55 fd0H1600
brw-rw----   1 root     floppy     2,  44 Aug  8 13:55 fd0H1680
brw-rw----   1 root     floppy     2,  60 Aug  8 13:55 fd0H1722
brw-rw----   1 root     floppy     2,  76 Aug  8 13:55 fd0H1743
brw-rw----   1 root     floppy     2,  96 Aug  8 13:55 fd0H1760
brw-rw----   1 root     floppy     2, 116 Aug  8 13:55 fd0H1840
brw-rw----   1 root     floppy     2, 100 Aug  8 13:55 fd0H1920
lrwxrwxrwx   1 root     root            7 Nov  1 15:39 fd0H360 -> fd0D360
lrwxrwxrwx   1 root     root            7 Nov  1 15:39 fd0H720 -> fd0D720
brw-rw----   1 root     floppy     2,  52 Aug  8 13:55 fd0H820
brw-rw----   1 root     floppy     2,  68 Aug  8 13:55 fd0H830
brw-rw----   1 root     floppy     2,   4 Aug  8 13:54 fd0d360
brw-rw----   1 root     floppy     2,   8 Aug  8 13:54 fd0h1200
brw-rw----   1 root     floppy     2,  40 Aug  8 13:54 fd0h1440
brw-rw----   1 root     floppy     2,  56 Aug  8 13:55 fd0h1476
brw-rw----   1 root     floppy     2,  72 Aug  8 13:55 fd0h1494
brw-rw----   1 root     floppy     2,  92 Aug  8 13:55 fd0h1600
brw-rw----   1 root     floppy     2,  20 Aug  8 13:54 fd0h360
brw-rw----   1 root     floppy     2,  48 Aug  8 13:55 fd0h410
brw-rw----   1 root     floppy     2,  64 Aug  8 13:55 fd0h420
brw-rw----   1 root     floppy     2,  24 Aug  8 13:54 fd0h720
brw-rw----   1 root     floppy     2,  80 Aug  8 13:55 fd0h880
brw-rw----   1 root     disk       3,   0 May  5  1998 hda
brw-rw----   1 root     disk       3,   1 May  5  1998 hda1
brw-rw----   1 root     disk       3,   2 May  5  1998 hda2
brw-rw----   1 root     disk       3,   3 May  5  1998 hda3
brw-rw----   1 root     disk       3,   4 May  5  1998 hda4
brw-rw----   1 root     disk       3,   5 May  5  1998 hda5
brw-rw----   1 root     disk       3,   6 May  5  1998 hda6
brw-rw----   1 root     disk       3,  64 May  5  1998 hdb
brw-rw----   1 root     disk       3,  65 May  5  1998 hdb1
brw-rw----   1 root     disk       3,  66 May  5  1998 hdb2
brw-rw----   1 root     disk       3,  67 May  5  1998 hdb3
brw-rw----   1 root     disk       3,  68 May  5  1998 hdb4
```

```
brw-rw----   1 root     disk       3,  69 May  5  1998 hdb5
brw-rw----   1 root     disk       3,  70 May  5  1998 hdb6
crw-r-----   1 root     kmem       1,   2 May  5  1998 kmem
crw-r-----   1 root     kmem       1,   1 May  5  1998 mem
lrwxrwxrwx   1 root     root           12 Nov  1 15:39 modem -> ttyS1
lrwxrwxrwx   1 root     root           12 Nov  1 15:39 mouse -> psaux
crw-rw-rw-   1 root     root       1,   3 May  5  1998 null
crwxrwxrwx   1 root     root      10,   1 Oct  5 20:22 psaux
brw-r-----   1 root     disk       1,   1 May  5  1998 ram
brw-rw----   1 root     disk       1,   0 May  5  1998 ram0
brw-rw----   1 root     disk       1,   1 May  5  1998 ram1
brw-rw----   1 root     disk       1,   2 May  5  1998 ram2
brw-rw----   1 root     disk       1,   3 May  5  1998 ram3
brw-rw----   1 root     disk       1,   4 May  5  1998 ram4
brw-rw----   1 root     disk       1,   5 May  5  1998 ram5
brw-rw----   1 root     disk       1,   6 May  5  1998 ram6
brw-rw----   1 root     disk       1,   7 May  5  1998 ram7
brw-rw----   1 root     disk       1,   8 May  5  1998 ram8
brw-rw----   1 root     disk       1,   9 May  5  1998 ram9
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 ramdisk -> ram0

***  我只把供我目前的 IDE partition 利用的設備包括進來。
***  如果你使用 SCSI，那麼請改用 /dev/sdXX 的設備。

crw-------   1 root     root       4,   0 May  5  1998 tty0
crw--w----   1 root     tty        4,   1 Nov  1 15:39 tty1
crw-------   1 root     root       4,   2 Nov  1 15:29 tty2
crw-------   1 root     root       4,   3 Nov  1 15:29 tty3
crw-------   1 root     root       4,   4 Nov  1 15:29 tty4
crw-------   1 root     root       4,   5 Nov  1 15:29 tty5
crw-------   1 root     root       4,   6 Nov  1 15:29 tty6
crw-------   1 root     root       4,   7 May  5  1998 tty7
crw-------   1 root     tty        4,   8 May  5  1998 tty8
crw-------   1 root     tty        4,   9 May  8 12:57 tty9
crw-rw-rw-   1 root     root       4,  65 Nov  1 12:17 ttyS1
crw-rw-rw-   1 root     root       1,   5 May  5  1998 zero

/etc:
-rw-------   1 root     root          164 Nov  1 15:39 conf.modules
-rw-------   1 root     root          668 Nov  1 15:39 fstab
-rw-------   1 root     root           71 Nov  1 15:39 gettydefs
-rw-------   1 root     root          389 Nov  1 15:39 group
-rw-------   1 root     root          413 Nov  1 15:39 inittab
-rw-------   1 root     root           65 Nov  1 15:39 issue
-rw-r--r--   1 root     root          746 Nov  1 15:39 ld.so.cache
-rw-------   1 root     root           32 Nov  1 15:39 motd
-rw-------   1 root     root          949 Nov  1 15:39 nsswitch.conf
drwx--x--x   2 root     root         1024 Nov  1 15:39 pam.d
-rw-------   1 root     root          139 Nov  1 15:39 passwd
-rw-------   1 root     root          516 Nov  1 15:39 profile
-rwx--x--x   1 root     root          387 Nov  1 15:39 rc
```

```
-rw-------   1 root     root           55 Nov  1 15:39 shells
-rw-------   1 root     root          774 Nov  1 15:39 termcap
-rw-------   1 root     root           78 Nov  1 15:39 ttytype
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 utmp -> ../var/run/utmp
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 wtmp -> ../var/log/wtmp

/etc/pam.d:
-rw-------   1 root     root          356 Nov  1 15:39 other

/lib:
-rwxr-xr-x   1 root     root        45415 Nov  1 15:39 ld-2.0.7.so
lrwxrwxrwx   1 root     root           11 Nov  1 15:39 ld-linux.so.2 -> ld-2.0.7.so
-rwxr-xr-x   1 root     root       731548 Nov  1 15:39 libc-2.0.7.so
lrwxrwxrwx   1 root     root           13 Nov  1 15:39 libc.so.6 -> libc-2.0.7.so
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libcom_err.so.2 -> libcom_err.so.2.0
-rwxr-xr-x   1 root     root         6209 Nov  1 15:39 libcom_err.so.2.0
-rwxr-xr-x   1 root     root       153881 Nov  1 15:39 libcrypt-2.0.7.so
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libcrypt.so.1 -> libcrypt-2.0.7.so
-rwxr-xr-x   1 root     root        12962 Nov  1 15:39 libdl-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libdl.so.2 -> libdl-2.0.7.so
lrwxrwxrwx   1 root     root           16 Nov  1 15:39 libext2fs.so.2 -> libext2fs.so.2.4
-rwxr-xr-x   1 root     root        81382 Nov  1 15:39 libext2fs.so.2.4
-rwxr-xr-x   1 root     root        25222 Nov  1 15:39 libnsl-2.0.7.so
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 libnsl.so.1 -> libnsl-2.0.7.so
-rwx--x--x   1 root     root       178336 Nov  1 15:39 libnss_files-2.0.7.so
lrwxrwxrwx   1 root     root           21 Nov  1 15:39 libnss_files.so.1 -> libnss_files-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libpam.so.0 -> libpam.so.0.64
-rwxr-xr-x   1 root     root        26906 Nov  1 15:39 libpam.so.0.64
lrwxrwxrwx   1 root     root           19 Nov  1 15:39 libpam_misc.so.0 -> libpam_misc.so.0.64
-rwxr-xr-x   1 root     root         7086 Nov  1 15:39 libpam_misc.so.0.64
-r-xr-xr-x   1 root     root        35615 Nov  1 15:39 libproc.so.1.2.6
lrwxrwxrwx   1 root     root           15 Nov  1 15:39 libpwdb.so.0 -> libpwdb.so.0.54
-rw-r-r---   1 root     root       121899 Nov  1 15:39 libpwdb.so.0.54
lrwxrwxrwx   1 root     root           19 Nov  1 15:39 libtermcap.so.2 -> libtermcap.so.2.0.8
-rwxr-xr-x   1 root     root        12041 Nov  1 15:39 libtermcap.so.2.0.8
-rwxr-xr-x   1 root     root        12874 Nov  1 15:39 libutil-2.0.7.so
lrwxrwxrwx   1 root     root           16 Nov  1 15:39 libutil.so.1 -> libutil-2.0.7.so
lrwxrwxrwx   1 root     root           14 Nov  1 15:39 libuuid.so.1 -> libuuid.so.1.1
-rwxr-xr-x   1 root     root         8039 Nov  1 15:39 libuuid.so.1.1
drwx--x--x   3 root     root         1024 Nov  1 15:39 modules
drwx--x--x   2 root     root         1024 Nov  1 15:39 security

/lib/modules:
drwx--x--x   4 root     root         1024 Nov  1 15:39 2.0.35

/lib/modules/2.0.35:
drwx--x--x   2 root     root         1024 Nov  1 15:39 block
drwx--x--x   2 root     root         1024 Nov  1 15:39 cdrom

/lib/modules/2.0.35/block:
drwx------   1 root     root         7156 Nov  1 15:39 loop.o
```

```
/lib/modules/2.0.35/cdrom:
drwx------   1 root     root        24108 Nov  1 15:39 cdu31a.o

/lib/security:
-rwx--x--x   1 root     root         8771 Nov  1 15:39 pam_permit.so

***  供 mount 時使用的 Directory stubs

/mnt:
drwx--x--x   2 root     root         1024 Nov  1 15:39 cdrom
drwx--x--x   2 root     root         1024 Nov  1 15:39 floppy

/proc:

/root:
-rw-------   1 root     root          176 Nov  1 15:39 .bashrc
-rw-------   1 root     root          182 Nov  1 15:39 .cshrc
-rwx--x--x   1 root     root          455 Nov  1 15:39 .profile
-rw-------   1 root     root         4014 Nov  1 15:39 .tcshrc

/sbin:
-rwx--x--x   1 root     root        23976 Nov  1 15:39 depmod
-rwx--x--x   2 root     root       274600 Nov  1 15:39 e2fsck
-rwx--x--x   1 root     root        41268 Nov  1 15:39 fdisk
-rwx--x--x   1 root     root         9396 Nov  1 15:39 fsck
-rwx--x--x   2 root     root       274600 Nov  1 15:39 fsck.ext2
-rwx--x--x   1 root     root        29556 Nov  1 15:39 getty
-rwx--x--x   1 root     root         6620 Nov  1 15:39 halt
-rwx--x--x   1 root     root        23116 Nov  1 15:39 init
-rwx--x--x   1 root     root        25612 Nov  1 15:39 insmod
-rwx--x--x   1 root     root        10368 Nov  1 15:39 kerneld
-rwx--x--x   1 root     root       110400 Nov  1 15:39 ldconfig
-rwx--x--x   1 root     root         6108 Nov  1 15:39 lsmod
-rwx--x--x   2 root     root        17400 Nov  1 15:39 mke2fs
-rwx--x--x   1 root     root         4072 Nov  1 15:39 mkfs
-rwx--x--x   2 root     root        17400 Nov  1 15:39 mkfs.ext2
-rwx--x--x   1 root     root         5664 Nov  1 15:39 mkswap
-rwx--x--x   1 root     root        22032 Nov  1 15:39 modprobe
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 reboot -> halt
-rwx--x--x   1 root     root         7492 Nov  1 15:39 rmmod
-rwx--x--x   1 root     root        12932 Nov  1 15:39 shutdown
lrwxrwxrwx   1 root     root            6 Nov  1 15:39 swapoff -> swapon
-rwx--x--x   1 root     root         5124 Nov  1 15:39 swapon
lrwxrwxrwx   1 root     root            4 Nov  1 15:39 telinit -> init
-rwx--x--x   1 root     root         6944 Nov  1 15:39 update

/tmp:

/usr:
drwx--x--x   2 root     root         1024 Nov  1 15:39 bin
```

```
drwx--x--x   2 root     root         1024 Nov  1 15:39 lib
drwx--x--x   3 root     root         1024 Nov  1 15:39 man
drwx--x--x   2 root     root         1024 Nov  1 15:39 sbin
drwx--x--x   3 root     root         1024 Nov  1 15:39 share
lrwxrwxrwx   1 root     root           10 Nov  1 15:39 tmp -> ../var/tmp

/usr/bin:
-rwx--x--x   1 root     root        37164 Nov  1 15:39 afio
-rwx--x--x   1 root     root         5044 Nov  1 15:39 chroot
-rwx--x--x   1 root     root        10656 Nov  1 15:39 cut
-rwx--x--x   1 root     root        63652 Nov  1 15:39 diff
-rwx--x--x   1 root     root        12972 Nov  1 15:39 du
-rwx--x--x   1 root     root        56552 Nov  1 15:39 find
-r-x--x--x   1 root     root         6280 Nov  1 15:39 free
-rwx--x--x   1 root     root         7680 Nov  1 15:39 head
-rwx--x--x   1 root     root         8504 Nov  1 15:39 id
-r-sr-xr-x   1 root     bin          4200 Nov  1 15:39 passwd
-rwx--x--x   1 root     root        14856 Nov  1 15:39 tail
-rwx--x--x   1 root     root        19008 Nov  1 15:39 tr
-rwx--x--x   1 root     root         7160 Nov  1 15:39 wc
-rwx--x--x   1 root     root         4412 Nov  1 15:39 whoami

/usr/lib:
lrwxrwxrwx   1 root     root           17 Nov  1 15:39 libncurses.so.4 -> libncurses.so.4.2
-rw-r-r---   1 root     root       260474 Nov  1 15:39 libncurses.so.4.2

/usr/sbin:
-r-x--x--x   1 root     root        13684 Nov  1 15:39 fuser
-rwx--x--x   1 root     root         3876 Nov  1 15:39 mklost+found

/usr/share:
drwx--x--x   4 root     root         1024 Nov  1 15:39 terminfo

/usr/share/terminfo:
drwx--x--x   2 root     root         1024 Nov  1 15:39 l
drwx--x--x   2 root     root         1024 Nov  1 15:39 v

/usr/share/terminfo/l:
-rw-------   1 root     root         1552 Nov  1 15:39 linux
-rw-------   1 root     root         1516 Nov  1 15:39 linux-m
-rw-------   1 root     root         1583 Nov  1 15:39 linux-nic

/usr/share/terminfo/v:
-rw-------   2 root     root         1143 Nov  1 15:39 vt100
-rw-------   2 root     root         1143 Nov  1 15:39 vt100-am

/var:
drwx--x--x   2 root     root         1024 Nov  1 15:39 log
drwx--x--x   2 root     root         1024 Nov  1 15:39 run
drwx--x--x   2 root     root         1024 Nov  1 15:39 tmp
```

```
/var/log:
-rw-------   1 root     root            0 Nov  1 15:39 wtmp

/var/run:
-rw-------   1 root     root            0 Nov  1 15:39 utmp

/var/tmp:
```

D 工具程式磁片 (utility disk) 目錄列表樣本 – Sample utility disk directory listing

```
total 579
-rwxr-xr-x   1 root     root        42333 Jul 28 19:05 cpio
-rwxr-xr-x   1 root     root        32844 Aug 28 19:50 debugfs
-rwxr-xr-x   1 root     root       103560 Jul 29 21:31 elvis
-rwxr-xr-x   1 root     root        29536 Jul 28 19:04 fdisk
-rw-r-r---   1 root     root       128254 Jul 28 19:03 ftape.o
-rwxr-xr-x   1 root     root        17564 Jul 25 03:21 ftmt
-rwxr-xr-x   1 root     root        64161 Jul 29 20:47 grep
-rwxr-xr-x   1 root     root        45309 Jul 29 20:48 gzip
-rwxr-xr-x   1 root     root        23560 Jul 28 19:04 insmod
-rwxr-xr-x   1 root     root          118 Jul 28 19:04 lsmod
lrwxrwxrwx   1 root     root            5 Jul 28 19:04 mt -> mt-st
-rwxr-xr-x   1 root     root         9573 Jul 28 19:03 mt-st
lrwxrwxrwx   1 root     root            6 Jul 28 19:05 rmmod -> insmod
-rwxr-xr-x   1 root     root       104085 Jul 28 19:05 tar
lrwxrwxrwx   1 root     root            5 Jul 29 21:35 vi -> elvis
```