Kernel HOWTO 中譯版

作者: Brian Ward *bri@blah.math.tu-graz.ac.at*
譯者: *C.W.Huang* & *Asd L. Chen*
v0.80, 26 May 1997. 翻譯日期: 25-29 August 1997

這是一份關於如何配置、編譯、升級核心以及排難解紛的詳細指引.

目錄

1 簡介

你是否需要閱讀這份文件? 如果你有下列任何症狀之一的話，是的：

- “哎呀！這套 wizzo-46.5.6 軟體說它需要 1.8.193 版的核心，而我卻仍然只有 1.0.9 版！”
- 比較新的核心之一裏面有你正好需要的一個設備驅動程式.
- 你對於如何編譯核心眞的一無所知.
- “在 README 檔案裡面眞的就是全部的資料了嗎?”
- 你來，你試，它還是不動.
- 你需要某事給予一直要求你爲他們安裝核心的人們.

1.1 請先看此!

這份文件中的某些範例假設你有 GNU tar，`find` 以及 `xargs`· 這些是 Linux 套件裡的標準工具，應該不是問題· 文件中也假設你知道你系統的檔案系統架構，如果你並不知道，最好趕快寫下一份 mount 指令在系統正常運作下，所顯示的結果作爲參考(或者是一份 /etc/fstab 的列表，如果你能夠看得懂)· 這些資訊很重要，而且，除非你重新分割或加入新的磁碟，重新安裝你的系統以及做諸如此類的操作，否則它們不會變動·

本文寫作時最新"產品"的核心版本號碼是 2.0.30，也就是說本文的參考及例子是對應於該版本的· 雖然我嘗試讓這篇文章儘量跟版本無關，核心卻不斷地在發展中· 因此如果你取得一新的版本，它不可避免地會有一些不同· 當然，這應該不會造成大問題，但它可能會製造一些混淆·

有兩種版本的 Linux 核心原始碼，"產品" 與 "發展中" 的· 產品版本從 1.0.x 開始而且目前是以偶數編號發表的; 1.0.x 是產品，1.2.x 是產品， 2.0.x 也是·這些版本應該比較穩定，在發表時是沒有臭蟲(bug)的版本· 發展中的核心 (1.1.x, 1.3.x 等等) 是作爲測試用的，給那些想要測試最新而且可能有許多臭蟲的人用的· 已經警告過你了·

1.2 文章風格說明

看起來像這樣 `text` 的文字或者是某樣將出現你的螢幕上的東西，一個檔名， 或是某樣可被直接鍵入的東西，像是命令，或命令的選項(如果你看的是純文字檔案，它看起來沒什麼不同)· 命令與其它的輸入經常被框起來(用' ')，這經常引起典型的標點符號問題: 如果這樣的項目出現在句子的最後面，人們時常會在命令後面加上一句號 '.'， 因爲美國人的引號習慣將句號放入引號裡面· 即使用常識(而不幸的是，這假設了擁有此"常識"的人是習慣於美國式的引號)想也知道應先將這標點去掉，但很多人總是忘記· 所以在此情形下我將句號放在引號外頭·換句話說，當我叫你要打 "`make config`" 時， 我會寫 '`make config`'，而不是 '`make config.`'·

2 重要問題與解答

2.1 不管怎麼樣，核心到底是做什麼的?

在 Unix 系統中，像是你的程式與硬體的一個仲裁者· 首先，它爲所有執行中的程式(程序)做記憶體管理， 並且確保它們都能夠平均(或不平均，如果你願意)的分享處理機的運算資源· 此外，它還提供了一個良好的介面讓剛剛提到的程式能透過它與你的硬體溝通·

其實核心所處理的工作比這還要更複雜一些，不過這些基本的功能是最必要知道的·

2.2 我爲什麼要更新核心?

比較新的核心一般來說會提供更強的能力來與更多種奇奇怪怪的硬體溝通(也就是說，它們擁有更多的設備驅動程式)， 它們能夠有更好的行程管理，它們能夠執行的比舊的版本更有效率，它們可能比舊的版本來的更穩定，而且它們修正了舊版裡的一些錯誤· 大部分的人是因爲他們需要那些新的設備驅動程式以及錯誤修正的部份而更新核心·

2.3 較新的核心支援那些硬體?

請看 *Hardware-HOWTO* 文件· 另一個方法是，你可以查看 Linux 原始程式碼中的 '`config.in`' 檔，

或者就在當你試著去 'make config' 時找出來· 'make config' 將會讓你看到標準的核心原始程式碼支援的所有硬體，但這並不是 Linux 所支援的全部硬體； 許多普通的設備驅動程式(像是 PCMCIA 驅動程式以及某些磁帶機的驅動程式)是個別維護及發行的可載入模組·

2.4 我需要那個版本的 gcc 或是 libc ?

Linus 對這個問題的建議放在 Linux 原始程式的 README 檔案裏頭· 如果你沒有該版本或是更新的版本，新版的 gcc 應該會告訴你是否需要更新 libc 的版本· 這兩個操作都不會有什麼問題，只要照著說明做就可以了·

2.5 什麼是可載入模組?

它們是核心的一部分(通常是設備驅動程式)，但是並沒有編譯到核心裡面去· 它們被分別編譯，然後幾乎可以在任何時候將它們插入運作中的核心或從中取出· 由於它的便利性，這已經成爲一種增加東西到核心裡去的較好方式· 許多常用的設備驅動程式，例如 PCMCIA 驅動程式以及 QIC-80/40 磁帶機的驅動程式就是可載入模組·

2.6 我需要多大的磁碟空間?

這與你系統的特殊配置有關·首先，壓縮過後的 Linux 原始程式碼在 2.0.10 版時約佔 6 MB· 許多站台在解壓縮後仍會保留一份·解壓縮以後這將佔掉約 24 MB· 但這還不是全部 — 你將會需要更多的磁碟空間來實際編譯這些東西· 這與你配置多少東西到你的核心裡頭去有關·例如，在某部機器上，我有網路，3Com 3C503 的驅動程式， 並且配置了三種檔案系統，而這全部共需要 30 MB·加上壓縮過的 linux 原始程式碼，這個配置大概需要用掉你 36MB · 在另外一個系統上，不需要支援網路設備(但仍然需要支援網路)，加上音效卡，結果吃掉更多空間· 還有，一個新版本的核心幾乎總是比舊版本的佔更大的空間· 所以，一般來說，如果你有許多種硬體設備的話，確定你有足夠的硬碟空間來容納這個配置· (以今日的價格來說，對於儲存空間的問題我不能不建議你再買一台磁碟機·)

2.7 這需要多久的時間?

對大部份的人而言，答案是"很久"·你的處理機速度有多快，以及你擁有的記憶體數量有多少將是決定所需時間的主要因素， 但是，在把多少東西配置到核心裡頭去這方面還是可以做點努力· 在一台有 16 MB 記憶體的 486DX4-100 電腦上，一個有五種檔案系統、支援網路、以及音效卡驅動程式的 1.2 版核心可以在二十分鐘以內完成· 類似的配置，在一台 386DX/40(8 MB 記憶體)上大約需要要 1.5 個小時· 在編譯核心時通常一些好的建議是去喝一點咖啡，看點電視，織些毛線，或任何你喜歡的事·如果你的機器眞的很慢的話你也該換台較快的機器 ·

3 如何眞正進行核心的配置?

3.1 取得原始程式碼

你可以經由匿名 ftp 站從 ftp.funet.fi 的 */pub/Linux/PEOPLE/Linus*， 或是其它 MIRROR 站取得·([譯註] 台灣可從 *ftp.edu.tw* 取得·) 檔案名稱一般是以標記成 linux-x.y.z.tar.gz 爲代表，其中的 x.y.z 是版本編號· 較新的(更好的?)版本以及修補檔一般是放在類似 'v1.1' 以及 'v1.2' 這樣的子目

錄下· 數字最大的是最新的版本，而且通常是"測試版"，也就是說如果你對 beta 或 alpha 版感到不安的話，你應該停留在主要的發行版本上·

強烈建議你去找 mirror 的 ftp 節點，而不要直接到 `ftp.funet.fi` 去! 底下是一些 mirrors 以及其他節點的簡要列表:

```
USA:        sunsite.unc.edu:/pub/Linux/kernel
USA:        tsx-11.mit.edu:/pub/linux/sources/system
UK:         sunsite.doc.ic.ac.uk:/pub/unix/Linux/sunsite.unc-mirror/kernel
Austria:    ftp.univie.ac.at:/systems/linux/sunsite/kernel
Germany:    ftp.Germany.EU.net:/pub/os/Linux/Local.EUnet/Kernel/Linus
Germany:    sunsite.informatik.rwth-aachen.de:/pub/Linux/PEOPLE/Linus
France:     ftp.ibp.fr:/pub/linux/sources/system/patches
Australia:  sunsite.anu.edu.au:/pub/linux/kernel
```

一般來說，`sunsite.unc.edu` 的 mirror 站是很好找的· 檔案 `/pub/Linux/MIRRORS` 包含了已知的 mirror 節點· 如果你沒有辦法上 ftp，有個存放 linux 的 BBS 系統列表會定期刊登在 comp.os.linux.announce 上，試著到那邊去找一下·

如果你想尋找一般的 Linux 資訊與套件，去 *http://www.linux.org/*·

3.2 解開原始程式碼的包裝

確定你是以'`root`'的身份簽入，然後 `cd` 到 `/usr/src`· 如果你安裝 Linux 時已經安裝了核心原始程式碼，在這個目錄底下應該已經存在一個叫做 '`linux`' 的子目錄· 這是舊版的 linux 原始程式碼·如果你還有磁碟空間而且想要玩得安全一點，那麼你會想要保留這個子目錄的· 有個好主意是根據你目前使用的核心版本來修改這個子目錄的名稱· '`uname -r`' 這個指令將會告訴你目前的核心版本· 所以，如果 '`uname -r`' 顯示 '`1.0.9`'，你就可以把 '`linux`' 改名(用 '`mv`')爲 '`linux-1.0.9`'· 如果你是個魯莽型的人，那就直接把這個子目錄清除掉· 無論是哪一種情況，反正只要確定在解開全部的原始程式碼之前，`/usr/src` 目錄下沒有 '`linux`' 這個子目錄就對了·

在 `/usr/src` 目錄下，用 '`tar zxpvf linux-x.y.z.tar.gz`' 來解開原始程式碼的包裝 (如果你拿到的是 `.tar` 的檔案，後面沒有加上 `.gz`，那就用 '`tar xpvf linux-x.y.z.tar`')· 你將會看到原始程式的內容飛逝而過·當它完成以後，在 `/usr/src` 將會出現一個新的 '`linux`' 子目錄· `cd` 到 `linux` 目錄下然後查看 `README` 檔， 裡面應該會有一段標題爲 '`INSTALLING the kernel`' 或類似的文字· 如果適當的話，先依照說明裡的指示 — 建立該有的符號鏈結，移除過時的 .o 檔案等等等諸如此類的工作·

3.3 配置核心

注意：這部份內容有些是重覆 Linus 的 `README` 檔案中的相應章節或加以修訂·

在 `/usr/src/linux` 下執行 '`make config`' 這個命令將會啓始一個指令稿， 而這個指令稿會問你各種問題·它需要 bash，所以要確定 bash 是 `/bin/bash`，`/bin/sh` 或 `$BASH`·

還有其它種 '`make config`' 的方式你可能會覺得更容易使用· 對那些執行 X Window 的人你如果有裝 Tk 的話可以試著用 '`make xconfig`'· '`make menuconfig`' 是給那些有裝 (n)curses 而且喜歡文字模式選單的人用的· 這些界面有個明顯的好處: 如果在配置過程中你不小心犯了錯，可以簡單地回去修正·

你現在該準備好回答這些問題，通常用 '`y`' 或 '`n`' 就可以了· 設備驅動程式通常會有一個 '`m`' 選項· 這表示"模組(module)"，表示說系統將會編譯它，但不會直接放進核心裡，而是做爲一可載入模組· 一

個有趣的説法是表示“maybe”· 有一些很明顯或是不重要的選項將不會在此加以説明· 請見“10 (其它配置選項)”一節對其中的簡短説明·

在 2.0.x 或以後的版本中有一個 ‘?’ 選項,提供了這個配置參數的簡短描述· 這些資訊應該都是最新的·

3.3.1 Kernel math emulation (核心的數學運算模擬)

如果你沒有數學輔助運算處理機(你只有一台單純的 386 或 486SX),那麼你在 這裡要回答 ‘y’· 如果你已經有數學輔助運算處理機卻還回答 ‘y’, 那也不必太擔心 — linux 還是會去使用它而忽略掉核心的模擬程式· 唯一的影響是編譯出來的核心變大了(用掉更多的記憶體)· 我聽説數學運簡模擬很慢; 雖然和本節無太大關係,還是應該注意這對 X Window 系統的效率會有很大影響·

3.3.2 Normal (MFM/RLL) disk and IDE disk/cdrom support (標準硬碟支援)

你在這裡差不多都得回答 ‘y’· 它代表的意思是核心將會支援標準的 PC 硬碟, 例如大多數人使用的 IDE 界面硬碟· 這不包括 SCSI 設備的支援; 這將在後面設定·

然後你會被問到關於 “old disk-only” 與 “new IDE” 驅動程式的事· 你應該兩者選一; 主要的差別是舊的驅動程式只支援單一界面兩部磁碟機, 而新的驅動程式支援第二界面及 IDE/ATAPI 光碟機· 新的驅動程式比舊的大約多 4K, 而且應該是“改良的”,也就是説除了可能包含些許錯誤之外,它應該會改善你的磁碟效率,特別是如果你有較新的 EIDE 硬體的話·

3.3.3 Networking support (網路支援)

原則上,如果你有連接網路,譬如説,你有 Internet 連線,或是你將使用 SLIP,PPP,term 或其它方式撥接到 Internet 上的話,回答 ‘y’· 然而,某些軟體套件(例如 X Window 系統)需要網路支援,即使你的機器並不眞正連接到網路上· 這情形你也應該回答 ‘y’· 然後,你會被問到是否需要支援 TCP/IP 網路,如果你不十分確定,就回答 ‘y’·

3.3.4 Limit memory to low 16MB (制記憶體在 16 MB)

可能會有一些不良的 386 DMA 控制器無法正確地定址到 16 MB 以上的記憶體位址; 如果你的機器剛好有這種(罕見)情況時,那麼你也就只好回答〞y〞了·

3.3.5 System V IPC

一個 IPC (行程內通訊,Interprocess Communication) 的定義就在 Perl 書中的註解· 這並不大奇怪,一些 Perl 程式設計者用它來讓行程之間互相溝通,就像許多其它套件(最著名如 DOOM)一樣· 因此除非你完全了解你在做什麼,不然最好別回答 n·

3.3.6 Processor type (386, 486, Pentium, PPro) (微處理機種類)

(在較舊版本的核心中爲: Use -m486 flag for 486-specific optimizations,使用 -m486 旗標針對 486 做最佳化)

按照慣例地，這是對一特別的微處理機做某種最佳化; 核心仍可在其它晶片上跑得很好，但可能會比較大些． 不過，在較新版本的核心上，已經不是這個樣子了．因此你應該正確地回答你要編譯的微處理機種類．“386” 的核心可以在所有的機器上工作．

3.3.7 SCSI support (SCSI 的支援)

如果你有 SCSI 設備，那麼就回答 ‘y’． 接著會有提示要求更進一步的資訊，像是你是否要支援光碟機，磁碟機，還有你使用的是那一種 SCSI 介面卡． 這部份請參閱 *SCSI-HOWTO*，有更詳細的說明．

3.3.8 Network device support (網路設備支援)

如果你有網路卡，或者你想要使用 SLIP，PPP，或是並列埠界面卡，那就回答 ‘y’． 接著會有提示問你使用的是那一種網路卡，或要使用那一種通訊協定．

3.3.9 Filesystems (檔案系統)

這將會有提示要你回答所要支援的檔案系統種類，計有：

Standard (minix) — 新的套件不再建立 minix 檔案系統，而且很多人不使用它，但是把它配置在核心裡仍然是個好主意． 某些“急救磁片”程式會用到它，而且仍然有許多軟碟磁片可能用 minix 檔案系統，因爲 minix 檔案系統對於處理軟碟片方面較無問題．

Extended fs — 這是擴充檔案系統的第一版，現在已經不再使用．使用的時機是，如果你有需要，你就會知道的．如果你覺得懷疑，你就不需要它．

Second extended — 這是現在新發行的套件所廣泛採用的檔案系統，你可能會有其中一種，所以必須回答 ‘y’．

xiafs filesystem — 這個檔案系統曾經一度很普遍，但是在寫這份文件時，我已經不知道有任何人在使用它了．

msdos — 如果你想要在 linux 下使用你硬碟中的 MS-DOS 分割區，或是想將用 MS-DOS 格式化的軟碟片掛進來的話，回答 ‘y’．

ums-dos — 這種檔案系統能使 MS-DOS 檔案系統擁有更多像 Unix 的特性，像是長檔名等等．這對那些不使用 MS-DOS 的人(像我)並不是很有用．

/proc — 另一種神奇的東西(我猜這個概念是無恥地從貝爾實驗室裡偷過來的)． 它不是你硬碟分割區裡的任何東西，而是核心與程序之間的檔案系統介面．許多程序工具(像 ‘ps’)都會用到它． 有時可以試著用 ‘`cat /proc/meminfo`’ 或 ‘`cat /proc/devices`’． 某些 shell (特別像是 rc) 使用了 `/proc/self/fd` (在其它系統裡叫做 `/dev/fd`) 來做輸出輸入動作． 你幾乎應該回答 ‘y’; 許多重要的 Linux 工具都依賴它來運作的．

NFS — 如果你在網路環境下而且想要由 NFS 來使用其它機器的檔案系統，回答 ‘y’．

ISO9660 — 絕大部分的光碟片都使用這個檔案系統．如果你有光碟機而且想在 Linux 下使用，回答 ‘y’．

OS/2 HPFS — 在編寫這份文件的時候，這個檔案系統還只支援到 OS/2 HPFS 的讀取．

System V and Coherent — 這是爲 System V 以及 Coherent 系統的分割區而設的(它們是另一種在 PC 上的 Unix 系統)．

但是我不知道我需要那些檔案系統! 吧，鍵入 'mount'· 它看起來會像這樣：

```
blah# mount
/dev/hda1 on / type ext2 (defaults)
/dev/hda3 on /usr type ext2 (defaults)
none on /proc type proc (defaults)
/dev/fd0 on /mnt type msdos (defaults)
```

仔細看看每一行；在 'type' 後面的那個字就是檔案系統的格式· 在這個例子中，我的 / 和 /usr 分割區是 second extended 格式， 我使用了 /proc，而且掛有一張以 msdos (bleah) 爲檔案系統格式的磁片·

如果你有使用 /proc，可以試試 'cat /proc/filesystems'· 它會給你一份目前使用的核心所支援的檔案系統列表·

這配置並不常使用·不重要的檔案系統會使核心顯得臃腫; 參見可載入模組一節的說明以避免此情況，以及"7 (陷阱)"一節說明爲何臃腫的核心不好·

3.3.10 Character devices (字元設備)

在這裡，你將會發現許多設備名稱，用來設定你的印表機，匯流排滑鼠，PS/2 滑鼠(大部分筆記型電腦使用內建的 PS/2 滑鼠)， 以及一些磁帶機驅動程式，以及其它的"字元"設備·當有合適的選項時，回答 'y'·

注意：Selection 這個程式使你可以在 X window 之外使用滑鼠在各個虛擬控制台間做剪貼的動作· 如果你有串列滑鼠的話，這眞的很棒，因爲它與 X window 之間的溝通良好， 不過如果你用其它種類的滑鼠就需要點技巧了·Selection 曾經一度是選擇的配置配項，不過現在已經是標準了·

注意二: Selection 現在已經過時了·現在新的程式名叫 "gpm"· 它能做更神奇的事，像是轉換滑鼠協定，處理多重滑鼠等……

3.3.11 Sound card (音效卡)

如果你想聽到 biff 程式大叫的話，回答 'y'· 然後等一下會編譯另一個配置程式並詳細詢問你所有關於音效卡的問題· (注意: 當它問你是否安裝完整版本的驅動程式時，你可以回答 'n' 以減少你核心記憶體並只選用你眞正需要的·) 如果你有音效卡我強烈建議看一下 *Sound-HOWTO* 以知道更多的細節·

3.3.12 其它配置選項

沒有將所有的選項都列出來，因爲有些經常改變，或是相當自明的(例如 3Com 3C509 support 就是編譯這種特別乙太網路卡的驅動程式)， 有一個十分容易理解的所有選項列表(包括將它們放到 Configure 指令稿的方法)可在這個 URL 找到(由 Axel Boldt *axel@uni-paderborn.de* 所收集):

http://math-www.uni-paderborn.de/~axel/config_help.html

或經由匿名 FTP 站:

ftp://sunsite.unc.edu/pub/Linux/kernel/config/krnl_cnfg_hlp.x.yz.tgz

其中 x.yz 是版本號碼·

其後來的核心中(2.0.x 及其後的)，這已被整合到原始程式碼中·

3.3.13 Kernel hacking

>這是從 Linus 的 README 裡摘錄的：

“kernel hacking” 配置的細節通常會產生一個更大或是更慢的核心(或者是又大又慢)， 而且可能因爲將一些常式配置成很活躍地去嘗試中斷一些不良的程式碼以便找出核心裡的問題（ kmalloc() ）， 而使得核心變得比較不穩定．所以要產生一個核心“產品”的話，你在這裡應該回答 ‘n’．

3.4 現在呢？(The Makefile)

在你完成 `make config` 之後，應該會有一個訊息告訴你核心已經配置好了， 以及要你去“check the top-level Makefile for additional configuration” 等等訊息．

所以查看一下 `Makefile`．你應該不必改變它，但是看看無妨． 一旦新核心完成，你也可以用 ‘`rdev`’ 指令來改變其中的選項．

4 編譯核心

4.1 清除與確認

當用來進行配置的指令稿執行完成時，它會告訴你去 ‘`make dep`’ 以及 ‘`clean`’． 所以要執行‘`make dep`’．這將確保所有的相依關係，例如 include files 都沒問題． 除非你的電腦眞的很慢，否則它不會花太久時間的． 完成後，在較舊版本的核心中你還應該做 ‘`make clean`’． 這會清除核心編譯的所有目的檔以及其它東西．在重建一個核心之前不要忘記這個步驟．

4.2 編譯時期

在完成 ‘`make dep`’ 及 ‘`make clean`’ 工作之後，你現在可以執行 ‘`make zImage`’ 或 ‘`make zdisk`’ (這部份需要長時間)． ‘`make zImage`’ 將會編譯核心，並且在 `arch/i386/boot` 留給你一個叫做 ‘`zImage`’ 的檔案．這就是新的壓縮核心． ‘`make zdisk`’ 做的事也一樣，但是它會把核心放到你所希望的磁片去，只要把這張磁片放在 “A:”． ‘`zdisk`’對於測試新核心很方便；如果它當掉(或工作不正常)，只要把磁片拿掉再用舊的核心啓動即可． 如果你意外地刪除了核心或是什麼的，它也是個方便的啓動辦法． 當你把一台磁碟的內容傾倒到另外一台去時，你也可以用它來安裝新的系統(除了這些之外還有更多用途！)

所有近來的核心都是壓縮過的，所以有個 ‘`z`’ 在名字前面．壓縮過的核心執行的時候會自動將自己解壓縮．

4.3 其他可以“make”的東西

‘`make mrproper`’ 將會做更廣泛的 ‘清除’ 工作． 這個動作有時候是必須的，所以你可能會希望在每次修補的時候執行它． ‘`make mrproper`’ 還會將你的配置檔案殺掉，所以如果你認爲它重要的話應該先做一備份(在 `.config`)．

‘`make oldconfig`’ 會嘗試由一舊的配置檔案來配置你的核心．它會爲你執行 ‘`make config`’． 如果你還未曾編譯過核心或沒有舊的配置檔案，那麼你可能不該做這個，因爲你幾乎確定會更改預設的配置．

參見可載入模組一節對 ‘`make modules`’ 的說明．

4.4 安裝核心

在你已經獲得一個看起來能夠照你希望運作的新核心之後，現在是安裝它的時候了． 大部份的人使用 LILO (Linux Loader) 來做這件事． ‘make zlilo’ 將會安裝核心，然後對它執行 LILO，使你完成所有開機的準備， 但是這只有在 LILO 已經以下列這種方式配置好的時候才可以：核心為 /vmlinuz， lilo 在 /sbin 下，而且與你的 LILO 配置(/etc/lilo.conf)一致．

另一方面，你可能需要直接使用 LILO．這是一個相當容易安裝的軟體，然而，它的配置檔常使人困擾． 查看配置檔(不是舊版的 /etc/lilo/config 就是新版的 /etc/lilo.conf)，看看裡面有些什麼．它看起來像這樣子的東西：

```
image = /vmlinuz
    label = Linux
    root = /dev/hda1
    ...
```

首先，‘image =’ 設定為已經安裝的核心．大部份的人似乎都用 /vmlinuz． ‘label’ 則是給 lilo 用來決定現在要啟動的是那個核心或作業系統， 而 ‘root’ 則是這個特別的作業系統的根目錄 /． 備份一份舊的核心，然後將你做的 zImage 拷貝進去(例如，如果你用 ‘/vmlinuz’ 的話，就像這樣 ‘cp zImage /vmlinuz’)． 然後，執行 ‘lilo’，但是在較舊的系統上，你可能必須執行 /etc/lilo/install 或甚至是 /etc/lilo/lilo -C /etc/lilo/config．

如果你想知道更多有關於 LILO 的配置，或是你並沒有 LILO 但是你想要安裝的話， 從你喜歡的 ftp 節點取得最新的版本然後依說明行事．

要能夠從硬碟啟動你的舊核心其中之一的話(這是另一個在新核心出狀況的情況下保護你自己的方法)， 把 LILO 配置檔中所有的(包括) ‘image = xxx’ 拷貝到檔案的最底端， 然後把 ‘image = xxx’ 改成 ‘image = yyy’ 其中的 ‘yyy’ 是你備份舊核心所存檔的名字． 接著，把 ‘label = zzz’ 改成像是 ‘label = linux-backup’ 然後重新執行 lilo． 你可能得要加上一行 ‘delay=x’，其中的 x 是以十分之一秒為單位的時間， 這是告訴 LILO 在啟動前先等一下，所以你可以中斷它的執行(例如用 shift 鍵)． 然後鍵入所備份的啟動映像之 label (在發生了某些令人不愉快的事情的情況下)．

5 修補核心

5.1 使用修補檔

核心的小幅更新是以修補檔的方式發行．例如，如果你的版本是 1.1.45， 而且你注意到出現了一個 ‘patch46.gz’ 可以用來升級它， 這代表你可以藉由使用這個修補檔將版本升級到 1.1.46． 你也許會想要先備份原始程式碼(‘make clean’ 然後 ‘cd /usr/src; tar zcvf old-tree.tar.gz linux’ 將會為你造出一份 tar 壓縮檔，裡面就是整個原始程式碼與其檔案架構)．

現在接著繼續上面的例子，假設你已經取得了 ‘patch46.gz’ 並放在 /usr/src 下． cd 到 /usr/src 然後執行 ‘zcat patch46.gz | patch -p0’ (如果並不是壓縮過的修補檔，那麼就執行 ‘patch -p0 < patch46’)， 你將會看到許多東西飛逝而過(或是慢慢跳過，如果你的電腦真的很慢的話)告訴你它正試著修補程式，以及是否成功． 通常，這個過程快得無法仔細看清楚它的動作，所以你會無法確定它到底有沒有成功地執行． 所以你可能會想在 patch 指令中加上 -s 選項，告訴 patch 程式只報告錯誤訊息就好． (這樣你就不會得到許多“嗨! 我的電腦正在做某些改變!”的感覺，不過你可能寧願這樣…) 那麼就得找找看一些可能不是那麼順利的事．cd 到 /usr/src/linux 並找尋檔名結尾為 .rej 的檔案．

某些版本的 patch（比較舊的版本，它們可能是在比較早期的檔案系統上編譯的）會把這些沒有成功的操作記錄在檔名有 # 的檔案裡． 你可以用 'find' 來替你找；

```
find . -name '*.rej' -print
```

會將現行目錄及子目錄下所有具 .rej 結尾的檔名印到標準輸出．

如果一切正確無誤的話，執行第三節跟第四節所討論的 'make clean'，'config' 以及 'dep'．

關於 patch 指令另外還有一些選項．如前所述，patch -s 將會抑制除了錯誤以外的所有其他訊息．如果你把核心原始程式碼放在 /usr/src/linux 以外的地方，在該目錄下用 patch -p1 也可以乾淨俐落的執行修補任務． 其它的 patch 選項在線上手冊(man page)裡都有很詳細的描述．

5.2 如果有錯誤發生

(注意: 本節提到的大部份是指舊版的核心)

通常最常發生的問題是當 patch 修改一個叫做 'config.in' 的檔案時，這個檔案看起來不怎麼對勁，因為你修改了一些選項來配合你的機器．這個問題已經小心的注意過了，但還是可能有人會在較舊的版本上遭遇這個問題． 要修正這個問題的話，查看 config.in.rej 這個檔案裡留下些什麼訊息． 有修改過的部份正常來說會在每一行開始的地方標上 '+' 和 '-'． 查看有標記的附近幾行，然後記得它們是設成 'y' 或是 'n'． 現在編輯 config.in 檔，然後在適當的地 方把 'y' 改成 'n' 以及 'n' 改成 'y'．執行

```
patch -p0 < config.in.rej
```

如果成功的話(沒有錯誤)，那麼你就可以繼續配置與編譯的工作． 這個 config.in.rej 檔還是會留在那裡，但是你可以砍掉它．

如果你仍然遭遇其它問題，那麼可能是你安裝了一份損壞的修補檔． 如果 patch 顯示 'previously applied patch detected: Assume -R?'， 你可能使用了一些比你目前的版本編號還低的修補檔; 如果你回答 'y'，它會嘗試將你的程式碼降級，而且幾乎一定會失敗; 如此，你就需要重新取得一份新的原始程式碼．

要還原到修補前的狀態的話，在原先的修補檔上執行 'patch -R'．

當修補檔真的失敗的時候，最好的辦法是從一個乾淨的原始程式碼(例如從 linux-x.x.x.tar.gz 檔案其中之一)開始，再重新執行．

5.3 砍掉 .orig 檔案

只要做過幾次修補以後，那些 .orig 的檔案將會開始堆積． 例如，我的一個 1.1.51 版的程式碼最後一次做清掃是在 1.1.48 版（我想是吧）． 刪除這些 .orig 檔案會節省許多的磁碟空間．

```
find . -name '*.orig' -exec rm -f {} ';'
```

將會替你照料這件事．某些版本的 patch 會用 ~ 來代替 .orig．

有其它更好的的辦法可砍掉這些 .orig 檔，就是利用 GNU 的 xargs:

```
find . -name '*.orig' | xargs rm
```

或是更安全但有些囉嗦的方法:

```
find . -name '*.orig' -print0 | xargs --null rm --
```

5.4 其它的修補檔

有一些並不是由 Linus 發行的其它修補檔出現(我將稱之爲"非標準的")． 如果你使用了它們，Linus 的修補檔可能會而無法正確地執行， 那麼你不是得將它們還原就是得因此而去修改原始程式碼或是修補檔．這個工作對初學者通常是很討厭， 所以如果你不想去修改原始程式碼(結果通常不大好)，在使用 Linus 的修補檔之前先還原這些非標準的修補檔，或重新安裝新的程式碼． 然後試試這非標準的修補檔能不能用．如果不能的話，那麼要不你就繼續使用舊的核心， 試著修改修補檔使其能用，要不就等待(可能是乞求)新版非標準修補檔出現．

非標準的修補檔有多普遍? 你有可能曾經聽過它們．我使用 Bill Paul 的不閃爍修補檔來修改我的虛擬控制台，因爲我討厭會閃爍的游標(這個修補檔經常爲新版核心而更新)． 由於大部份較新的設備驅動程式都已經被發展成可載入模組形式，因此許多非標準的修補檔的重要性已逐漸降低．

6 附加的套件

你的 Linux 核心有許多在原始程式碼本身裡面並沒有說明的特性； 這些特性一般是經由外來的軟體來利用，在這裡列出一部分最普遍的：

6.1 kbd

Linux 的控制台有著比你所能吃驚更多的特色．這包括切換字型，重新對映你的鍵盤，切換顯示模式(較新版的核心)的能力等等． kbd 這套軟體裡有能夠讓使用者做這些動作的支援程式，還加上一大堆的字型以及幾乎足以適用任何鍵盤的一些鍵盤對映表． 它可在放置核心原始碼的同一站台找到．

6.2 util-linux

Rik Faith (*faith@cs.unc.edu*) 收集了一大堆 Linux 的工具，叫做 util-linux． 現在是由 Nicolai Langfeldt (*util-linux@math.uio.no*) 所維護． 可從 sunsite.unc.edu 的 */pub/Linux/system/misc* 取得． 它包括了像 `setterm`，`rdev` 以及 `ctrlaltdel` 與核心有關的工具． 就像 Rik 所說的，不要想都不想就把它裝上去*!* 你不需要安裝此套件中的每一個東西，而且如果你這樣做的話可能會引起嚴重的問題．

6.3 hdparm

像很多軟體一樣，這曾經是一個核心修補檔及其支援程式． 這些修補檔被放入商業核心中，而用來最佳化以及調適你硬碟的支援程式一般是分開發行的．

6.4 gpm

gpm 是 general purpose mouse 的簡寫． 這個程式可以讓你使用不同種類的滑鼠在虛擬控制台之間剪貼，以及做一些其它的事情．

7 一些陷阱

7.1 make clean

如果你的新核心會做一些眞的很奇怪的事，有可能是因爲在編譯核心前你忘了做清除 make clean · 症狀從你的核心不正常地崩潰到奇怪的輸出入問題，一直到可憐的執行效率等等不一而足，可以是任何事· 最好也要確定你有做 make dep ·

7.2 巨大或緩慢的核心

如果你的核心佔用了大量的記憶體，或者它眞的是很大很大，也或者是即使用你全新的 786DX6/440 來編譯卻都還像是永遠編譯不完的話， 那麼有可能是因爲你配置了太多不必要的東西(設備驅動程式，檔案系統等等)· 如果你不會用到某些東西，那就不要配置它，因爲它眞的會佔用記憶體· 核心過於臃腫最明顯的症狀就是發生記憶體與磁碟之間異常大量的資料交換· 如果你不是用那種聲音聽起來好像是噴射機降落的舊型 Fujitsu Eagles 硬碟，檢查一下你的核心配置·

你可以找出你機器上全部記憶體的數量，然後減掉 /proc/meminfo 裡面的 “total mem” 或 ‘free’ 指令所得的記憶體數量來得知核心使用了多少記憶體· 你也可以執行 ‘dmesg’ (或者也可以查看核心的記錄檔，它一定在會你的系統裡)·看起來就像這一行：

```
Memory:  15124k/16384k available (552k kernel code, 384k reserved, 324k data)
```

我的 386 (配置很少垃圾)顯示如下：

```
Memory:  7000k/8192k available (496k kernel code, 384k reserved, 312k data)
```

如果你‘必須’得到一大型核心但系統卻不讓你做，你可以試試 ‘make bzimage’· 你可能必須安裝新版的 LILO 來做這件事·

7.3 核心無法編譯

如果它沒有被編譯，那麼可能是有個修補檔失敗了，或者是你從某個地方拿到的原始程式碼有問題·也有可能是因爲你的 gcc 版本不正確或壞掉了(例如含入檔有錯誤)· 確定 Linus 在 README 裡所描述的符號鏈結都有正確建立· 一般說來，如果核心沒能編譯，這表示在某些地方有嚴重的錯誤，重新安裝某些工具可能是必須的·

或者可能你用 ELF 編譯器 (gcc 2.6.3 或以後的) 來編譯 1.2.x 的核心· 如果編譯過程中你得到一大堆的 xxxx undefined 的訊息，這可能是你的問題· 修正的方法大部份都很簡單·將這幾行加到 arch/i386/Makefile 的頂端: arch/i386/Makefile:

```
AS=/usr/i486-linuxaout/bin/as
LD=/usr/i486-linuxaout/bin/ld -m i386linux
CC=gcc -b i486-linuxaout -D__KERNEL__ -I$(TOPDIR)/include
```

然後重新執行 make dep 與 zImage ·

在少數情況下，gcc 可能會由於硬體問題而當掉·錯誤訊息會像 “xxx exited with signal 15” 之類的，而且會看起來很奇怪· 我本來不想提這點的，不過在我身上也發生過一次 - 我有一些壞的 cache 記憶體，編譯器時常會隨機地當掉· 如果你有此問題的話先試著重新安裝 gcc ·如果你將外部 cache 關掉，減少一些 RAM 之後核心就編譯成功了，你大概只會覺得可疑·

告訴人們他的硬體有問題常會使人困擾．不過，這不是我發明的． 這是一個 FAQ – 可以在 *http://www.bitwizard.nl/sig11/* 找到．

7.4 新版的核心似乎不能啓動

你沒有執行 LILO，或是沒有正確的配置它．有一次我曾經碰到的問題是出在配置檔裡， 我用了 'boot = /dev/hda1' 而不是 'boot = /dev/hda' (這在剛開始時眞的是很討厭，但是一旦你有了一個可以用的配置檔，應該不需要去再去改變它)．

7.5 你忘了執行 LILO，或系統根本不能啓動

噢！現在最好的辦法是用磁片啓動，並且準備另一張可以啓動的磁片(像是'make zdisk'時做的磁片)．你得知道你的根目錄(/)所在的分割區以及它的格式(second extended, minix 等等)． 在下面的例子中，你也得知道你的 /usr/src/linux 原始程式碼在那個分割區，它的格式，以及它一般會掛在那兒．

在這個例子中， 根目錄 / 是 /dev/hda1，而持有 /usr/src/linux 的分割區是 /dev/hda3，一般會掛在 /usr 下． 它們都是 second extended 檔案系統．可以運作的核心映像叫做 zImage，放在 /usr/src/linux/arch/i386/boot 底下．

這個主意是這樣的，假若有一個可以運作的核心映像叫做 zImage，可能可以把它用在新的磁片上． 另外一個不一定會更好的變通辦法(這跟你的系統怎麼組成的有關)在說明這個例子之後會討論到．

首先，從 boot/root 磁片或者是急救磁片開機，然後將持有可運作核心的分割區掛上來：

```
mkdir /mnt
mount -t ext2 /dev/hda3 /mnt
```

如果 mkdir 指令顯示該目錄已經存在，忽略掉不必理會它． 現在，cd 到持有可運作核心的地方．注意:

```
/mnt + /usr/src/linux/arch/i386/boot - /usr = /mnt/src/linux/arch/i386/boot
```

把一張格式化過的磁片放進 "A:" 磁碟機(確定不是你的 boot/root 磁片!)， 把映像檔傾倒到磁片裡去，然後配置你的根目錄分割區:

```
cd /mnt/src/linux/arch/i386/boot
dd if=zImage of=/dev/fd0
rdev /dev/fd0 /dev/hda1
```

cd 到根目錄 / 並且卸下標準 /usr 分割區:

```
cd /
umount /mnt
```

你現在應該可以從這張磁片正常的開機了．在這次開機後不要忘記執行 lilo (或是其它你曾經做錯的什麼事)!

如同前面曾經提過的，還有另外一種很普遍的變通方式． 如果情況是你有一個可以運作的核心在放在 / (例如 /vmlinuz)，你也可以使用它． 假定所有的條件都跟上面的例子一樣，而我的核心

映像是 /vmlinuz，只要對上面的例子做這些改變: 把 /dev/hda3 改成 /dev/hda1 (/ 分割區)， 把 /mnt/src/linux 改成 /mnt，並且把 if=zImage 改成 if=vmlinuz． 至於前面有關注意如何推導出 /mnt/src/linux/arch/i386/boot 的那個部分可以忽略．

將 LILO 使用在大的硬碟上(超過 1024 磁柱)可能會有問題． 請參見 LILO mini-HOWTO 或其它文件的說明．

7.6 系統表示 'warning: bdflush not running'

這可以算是一個相當嚴重的問題．從 1.0 版以後的核心開始(大概是在 1994 年四月二十日左右)， 有個會週期性地更新檔案系統緩衝區的程式叫做 'update' 被升級或取代掉了． 取得 'bdflush' 的原始程式碼(你應該可以從你取得核心的地方找到)， 然後編譯它(你可能會希望在舊版的核心下執行編譯及安裝)． 它會以 'update' 爲名安裝它自己並且在重開機以後，新核心應該會運作良好．

7.7 系統說 undefined symbols 而且無法編譯

你可能有一 ELF 編譯器(gcc 2.6.3 或以後的)而且是 1.2.x (或更早的)核心原始碼． 一般修正的方法是將這幾行加到 arch/i386/Makefile 的頂端:

```
AS=/usr/i486-linuxaout/bin/as
LD=/usr/i486-linuxaout/bin/ld -m i386linux
CC=gcc -b i486-linuxaout -D__KERNEL__ -I$(TOPDIR)/include
```

這會以 a.out 程式庫來編譯 1.2.x 核心．

7.8 無法讓我的 IDE/ATAPI CD-ROM 正常工作

很奇怪，一大堆人無法讓他們的 ATAPI 光碟機工作，可能是因爲有太多事容易出錯．

你的光碟機是在一特別 IDE 界面上的唯一設備，它必須被調整爲 "master" 或 "single"． 這可能是最常見的錯誤．

Creative Labs 現在將 IDE 界面放到他們音效卡裡． 然而，這將導致一個有趣的問題，雖然有些人只有一個 IDE 界面，許多人在主機板上有兩個內建的 IDE 界面(通常在 IRQ15)， 因此一解決的辦法是將聲霸卡的界面調成第三個 IDE (有人告訴我是 IRQ11)．

這在 1.2.x 的 Linux 核心上會有問題，因爲它不支援第三個 IDE 界面(從 1.3.x 系列已開始支援，但它還在發展中，而且不會自動偵測)． 要解決此問題，你有一些選擇．

如果你已經有第二個 IDE 埠，如果你沒用它或沒有兩部設備在上面的話就有機會． 將 ATAPI 光碟機從音效卡上拿下來並放到第二個界面上．然後你可以關掉音效卡上的界面，這就可以省下一個 IRQ．

如果你沒有第二個 IDE 界面，調整音效卡的界面(不是音效卡的音效部份)到第二界面用的 IRQ15，這樣應該會動．

如果因爲某些理由非得使用"第三個"界面不可，或是有其它問題， 取得 1.3.x 的核心(例如 1.3.57 就有)，閱讀 drivers/block/README.ide 檔案． 那裡有更多的資訊說明．

7.9 系統顯示關於 obsolete routing requests 的奇怪訊息

取得新版的 route 程式及其它與 route 有關的程式． /usr/include/linux/route.h (這是 /usr/src/linux 下的一個檔案)已經做了修改．

7.10 防火牆功能無法在 1.2.0 上工作

至少升級到 1.2.1 版．

7.11 "Not a compressed kernel Image file" (非壓縮核心映像檔)

不要用在 /usr/src/linux 產生的 vmlinux 做爲你的啓動核心映像; [..]/arch/i386/boot/zImage 才是正確的．

7.12 升級至 1.3.x 後在控制台終端機上的問題

將控制台設定檔 /etc/termcap 中的 dumb 改爲 linux． 你可能會必須增加一項 terminfo．

7.13 核心升級後似乎無法編譯東西

Linux 的核心原始程式碼包含了許多的含入檔(就是用 .h 結尾的檔案)必須爲標準的 /usr/include 所參考． 它們通常用這種方法被參考(其中 xyzzy.h 是在 /usr/include/linux 下):

```
#include <linux/xyzzy.h>
```

正常情況下，在 /usr/include 下會有一叫做 linux 的連結到你的核心原始碼的 include/linux 目錄(一般系統在 /usr/src/linux/include/linux)． 如果這個連結沒有了，或指到錯誤的地方，大部份的東西都將無法編譯． 如果你覺得核心原始碼佔了太多的空間而砍掉它，這顯然會引發問題． 另一個可能的錯誤是它的檔案權限; 如果你的 root 預設不讓其它使用者看到他的檔案， 而且你解開核心原始碼時沒有加上 p (保留檔案模式)選項，其它使用者也會無法使用 C 編譯器． 雖然你可以用 chmod 指令來修正，不過更容易的方法是重新解開含入檔． 你可以一開始你解開整個原始碼的同樣方法，不過多加了一個參數:

```
blah# tar zxvpf linux.x.y.z.tar.gz linux/include
```

請注意: 如果 /usr/src/linux 連結不在的話 "make config" 會重建之．

7.14 增加上限

下面一些範例指令告訴你如何增加核心提供的上限:

```
echo 4096 > /proc/sys/kernel/file-max
echo 12288 > /proc/sys/kernel/inode-max
echo 300 400 500 > /proc/sys/vm/freepages
```

8 升級至 2.0.x 的注意事項

核心 2.0.x 版在安裝上引入了許多的改變. 在原始程式碼裡的 Documentation/Changes 檔案包含了升級至 2.0.x 版所應知道的資訊. 你可能還需要升級一些關鍵的套件，例如 gcc，libc 與 SysVInit，並改變一些系統檔案. 不過，也別緊張.

9 可載入模組

可載入模組能夠節省記憶體，而且很容易設定. 模組的範圍已經包含了檔案系統，乙太網路卡驅動程式，磁帶機，印表機驅動程式，還有更多.

9.1 安裝模組工具

模組工具可以從你取得核心的地方找到，如 modules-x.y.z.tar.gz; 選擇與你目前核心版本相等或稍低的最接近的 x.y.z. 用 'tar zxvf modules-x.y.z.tar.gz' 解開. cd 到它產生的目錄(modules-x.y.z)，看一下 README，然後按照安裝指示執行 (通常很簡單，就像 make install 之類的). 然後你在 /sbin 應該會有這些程式 insmod, rmmod, ksyms, lsmod, genksyms, modprobe, 以及 depmod. 如果你想的話，可以用 insmod 裡的範例驅動程式"hw"來測試這些工具; 參見 INSTALL 檔裡的細節說明.

insmod 指令將一個模組插入運行中的核心. 模組通會以 .o 爲結尾; 上面提到的範例程式就叫做 drv_hello.o，因此要插入此模組，就用 'insmod drv_hello.o'. 要了解目前核心正在使用的模組，用 lsmod 指令. 輸出看起來像這樣:

```
blah# lsmod
Module:        #pages:  Used by:
drv_hello          1
```

'drv_hello' 是這模組的名稱，它用了一頁 (4K) 的記憶體，而且目前沒有其它的核心模組依賴於它. 要移除此模組，用 'rmmod drv_hello'. 注意 rmmod 需要的是模組名稱，而不是檔案名稱; 就是你從 lsmod 指令列出來的. 其餘工具的作用在線上手冊中都有說明.

9.2 與核心發行的模組

在 2.0.30 版中，幾乎所有的東西都可以變成可載入模組. 要使用可載入模組，首先確定你沒有將其配置到正常的核心裡，也就是說，在 'make config' 時不是回答 y. 編譯新核心並用它來開機. 然後，再次 cd 到 /usr/src/linux，然後執行 'make modules'. 這將編譯所有你未指定在核心配置裡的模組，並將其連結到 /usr/src/linux/modules. 你可以直接由此目錄使用，或是執行 'make modules_install' 將其安裝到 /lib/modules/x.y.z，其中 x.y.z 是核心版本.

這對處理檔案系統上特別便利. 你可能並不經常使用 minix 或 MS-DOS 的檔案系統. 例如，如果我遇到一 MS-DOS 的軟碟片，我會用 insmod /usr/src/linux/modules/msdos.o，並在使用完成後用 rmmod msdos 指令將其去掉. 此過程大約可以節省核心 50K 左右的記憶體. 對於 minix 檔案系統有一點要注意的就是， 爲了要使用"急救磁片"，你應該總是直接將它配置到核心中.

10 其它配置選項

本節包括了在配置一節中未列出的核心配置選項(在 `make config` 時)· 大部份的設備驅動程式並未列出·

10.1 General setup (一般設定)

`Normal floppy disk support` (一般軟碟支援) - 就是這個· 你可能會想看一下 `drivers/block/README.fd` 這個檔案; 對 IBM Thinkpad 的使用者來說特別重要·

`XT harddisk support` (XT 硬碟支援) - 如果你想用老古董的八位元 XT 控制器的話·

`PCI bios support` (PCI bios 的支援) - 如果你有 PCI，你可能會想打開此選項· 但要小心，某些舊的 PCI 主機板在此選項下可能會當掉·更多關於 PCI 匯流排在 Linux 下使用的資訊可以在 *PCI-HOWTO* 裡找到·

`Kernel support for ELF binaries` (核心支援 ELF 執行檔) - ELF 是一種新的二進位執行檔· Linux 似乎正朝此方向前進，所以你可能會要這個·

`Set version information on all symbols for modules` - 在過去，核心模組與每個新核心一起被編譯· 如果你回答 y，它可能讓你使用不同 patchlevel 下編譯的模組· 參見 `README.modules` 中的詳細說明·

10.2 Networking options (網路選項)

網路選項在 *NET-3-HOWTO* (或是 NET-什麼-HOWTO)中描述·

11 祕訣和技巧

11.1 重導 make 或是 patch 指令的輸出

如果你想要記錄這些 '`make`' 或是 '`patch`' 指令到底做了些什麼， 你可以把螢幕的輸出重導到一個檔案去·首先，要知道你用的是是那一種 shell： '`grep root /etc/passwd`' 然後尋找看起來像 '`/bin/csh`' 一類的東西·

如果你使用的是 sh 或是 bash，

```
(command) 2>&1 | tee (output file)
```

將會把一份 `(command)` 的輸出放到 '`(output file)`' 這個檔案去·

如果是 csh 或 tcsh，用

```
(command) |& tee (output file)
```

這個語法·

如果是 rc (注意：你應該不會是用 rc)，語法是

```
(command) >[2=1] | tee (output file)
```

11.2 條件式的核心安裝

除了使用軟碟片，還有數種在不動舊核心的情況下測試新核心的方法． 不像其它的 Unix 系統，LILO 具有從任何地方啓動核心的能力． (不過如果你有大的硬碟，超過 500 MB 的，請參考 LILO 的文件說明將可能引起的問題．) 因此，如果你在 LILO 配置檔案中用像這樣的

```
image = /usr/src/linux/arch/i386/boot/zImage
    label = new_kernel
```

你可以在不動舊核心 `/vmlinuz` 的情況下選擇執行新編譯的核心(當然，在執行完 `lilo` 後)． 叫 LILO 啓動新核心最簡單的方法就是在啓動時按下 shift 鍵(當你在螢幕只看到 `LILO` 時)， 這會給你一提示符號．此時你可以鍵入 ‘`new_kernel`’ 來啓動新核心．

如果你想在系統中同時保有數種不同的核心原始程式碼(小心，這會用掉大量的硬碟空間)， 最普通的方法就是將其命名爲 `/usr/src/linux-x.y.z`，其中 `x.y.z` 是核心版本． 然後你可以用符號連結來“選擇”一份原始碼; 例如，用 ‘`ln -sf linux-1.2.2 /usr/src/linux`’ 會讓 1.2.2 成爲目前的核心樹． 但在建立符號連結之前，先確定 `ln` 指令最後一個參數不是一個眞正的目錄(舊的符號連結沒有關係); 否則結果可能不會是你所預期的．

11.3 核心更新

Russell Nelson (*nelson@crynwr.com*) 總結了新版核心的改變． 這些是簡短的，而且你在升級前可能應該先看一下．可以從匿名 FTP 站 `ftp.emlist.com` 的 *pub/kchanges*，

12 其它可能相關有用的 HOWTOs

- Sound-HOWTO: 音效卡與工具
- SCSI-HOWTO: 所有關於 SCSI 控制卡與設備的資訊
- NET-2-HOWTO: 網路
- PPP-HOWTO: 特別關於 PPP 網路的
- PCMCIA-HOWTO: 筆記型電腦的驅動程式
- ELF-HOWTO: 什麼是 ELF，如何轉換等…
- Hardware-HOWTO: 支援硬體的概要
- Module-HOWTO: 更多與核心模組有關的
- Kerneld mini-HOWTO: 關於 kerneld
- BogoMips mini-HOWTO: 你可能會有興趣的

13 雜項

13.1 作者

Linux Kernel-HOWTO 的作者及維護人是 Brian Ward (*bri@blah.math.tu-graz.ac.at*)· 請將任何評論，補充與更正寄給我(更正對我來說是特別重要的)·

可以從這些 URL 找到我的網頁:

http://www.math.psu.edu/ward/
http://blah.math.tu-graz.ac.at/~bri/

雖然我儘可能地去處理電子郵件，但請記住我每天都收到大量的信件，所以可能要花一點時間才會回覆你· 特別是寄信來問我問題的，請儘量非常清楚與詳盡地描述你的訊息· 如果你說的是某個無法運作的硬體(或類似的)，我需要知道你的硬體配置是什麼· 如果你要回報錯誤，不要只是說"我這樣試但它給我會錯誤訊息"; 我需要知道這個錯誤訊息是什麼· 我可能也想知道你的核心，gcc，libc 的版本是什麼，如果你只是告訴我你用的是這個或那個套件，那沒什麼用· 我不介意你問我簡單的問題; 請記住，如果你不問，你將永遠不會得到解答· 我要感謝每個給我回饋的人們·

如果你寫信給我但在一段合理的時間內沒有得到回應(三星期或更久)，那麼可能我不小心砍掉了你的信件或諸如此類的(抱歉)· 請再試一次·

我收到許多信件是關於硬體問題的·這沒問題，但請了解我並不是熟悉世界上所有硬體的人而且我也不知道我能幫多少忙· 我個人使用的機器有 IDE 與 SCSI 硬碟，SCSI 光碟機，3Com 與 WD 的乙太網路卡，串列滑鼠， PCI 主機板，NCR 810 SCSI 控制器，AMD 386DX40 w/Cyrix 輔助處理器，AMD 5x86，AMD 486DX4 與 Intel 486DX4 處理器 (這只是我在使用而且熟悉的硬體摘要，並不是建議，但如果你想要用，歡迎來詢問 :-))·

0.1 版完成於 1994 年十月三日·本文件可以 SGML, PostScript, TeX, roff 與純文字格式取得·

13.2 預定目標

"祕訣和技巧" 一節太少了·我希望能用從別人得到的建議來擴充·

"附加的套件" 一節也是·

還需要更多的偵錯/當機回復資訊·

13.3 貢獻者

包含了一小部份 Linus 的 README 檔案 (kernel hacking 選項)·感謝 Linus!

`uc@brian.lunetix.de` (Ulrich Callmeier): patch -s and xargs.

`quinlan@yggdrasil.com` (Daniel Quinlan): corrections and additions in many sections.

`nat@nat@nataa.fr.eu.org` (Nat Makarevitch): mrproper, tar -p, many other things

`boldt@math.ucsb.edu` (Axel Boldt): collected descriptions of kernel configuration options on the net; then provided me with the list

`lembark@wrkhors.psyber.com` (Steve Lembark): multiple boot suggestion kbriggs@earwax.pd.uwa.edu.au (Keith Briggs): some corrections and suggestions

rmcguire@freenet.columbus.oh.us (Ryan McGuire): makeables additions

dumas@excalibur.ibp.fr (Eric Dumas): French translation

simazaki@ab11.yamanashi.ac.jp (Yasutada Shimazaki): Japanese translation

jjamor@lml.ls.fi.upm.es (Juan Jose Amor Iglesias): Spanish translation

mva@sbbs.se (Martin Wahlen): Swedish translation

jzp1218@stud.u-szeged.hu (Zoltan Vamosi): Hungarian translation

bart@mat.uni.torun.pl (Bartosz Maruszewski): Polish translation

donahue@tiber.nist.gov (Michael J Donahue): typos, winner of the "sliced bread competition"

rms@gnu.ai.mit.edu (Richard Stallman): "free" documentation concept/distribution notice

dak@Pool.Informatik.RWTH-Aachen.DE (David Kastrup): NFS thing

esr@snark.thyrsus.com (Eric Raymond): various tidbits

曾寄信給我問問題的人也有很大幫助.

13.4 版權聲明，許可，諸如此類的

Recommendations: Commercial redistribution is allowed and encouraged; however, it is strongly recommended that the redistributor contact the author before the redistribution, in the interest of keeping things up-to-date (you could send me a copy of the thing you're making while you're at it). Translators are also advised to contact the author before translating. The printed version looks nicer. Recycle.