第六章

檔案系統管理

大概在第一次安裝 Linux 時，檔案系統和置換空間就已經建立好了（大多數的套件都會幫你完成這些基本的工作）。現在機會來了！你還可以進一步調整這些資源；通常會在作業系統剛安裝完成，還未將一些好玩的東西放進硬碟前來做這件事，但偶爾你會為系統加上新的裝置，或者是在增加 RAM 的同時也需要增加置換空間，這時就必需改變已在運作中的檔案系統。

管理檔案系統

對 UNIX 來說，檔案系統是某種經過格式化之後可以儲存檔案的裝置（例如：硬碟、軟碟或光碟片）。檔案系統可以放在硬碟、軟碟、光碟和其它可以隨機存取的儲存媒介（磁帶只能循序存取，所以磁帶就本質上來說是不能包含真正的檔案系統的）。

究竟用什麼格式或什麼方法來儲存檔案並不重要；系統會提供一個共通的界面給所有能被辨識的檔案系統類型。在 Linux 下，檔案系統的類型包括「第二延伸檔案系統」（Second Extended Filesystem）或稱為 *ext2fs*，或許你就是用它來儲存 Linux 檔案（還有 *ext3* 也正逐漸地變得通行）；還有 VFAT 檔案系統，可在 Linux 下存取 Windows 95/98/ME 分割區與軟碟中的檔案；此外，Linux 也提供了光碟機所用的 ISO 9660 等等類型的檔案系統。

各種檔案系統都有一套大異其趣的儲存資料格式來作為基礎，然而，在 Linux 下使用檔案系統時，系統則是以目錄結構的檔案形式來呈現這些資料，而且會附帶一些熟悉的屬性，如：檔案所有者、群組識別碼、使用權限位元（permission bit）...等。

實際上，這些檔案所有者、使用權限等屬性資訊，只有在那些原本就是為儲存 Linux 檔案而設計的原生 Linux 檔案系統上才會真正有提供。對於實際上並未儲存這類資訊的檔案系統而言，核心中存取這些檔案系統的驅動程式會「捏造」出這些屬性。例如，MS-DOS 檔案系統根本就沒有所謂的「所有者」這種概念，所以，MS-DOS 檔案系統的驅動程式會假裝所有的檔案都是由 *root* 所擁有。這麼一來，就某個層次而言，所有類型的檔案系統看來都是一樣的，而每個檔案都伴隨有一些相關的屬性，至於在下面的檔案系統到底是否真得用到了這些資訊，則又完全是另外一回事情。

身為一個系統管理者，如果想把 Linux 的檔案儲存在軟碟片上，或是想在硬碟上增設額外的檔案系統，則必須知道如何建立檔案系統；如果資料損毀了，也必須知道如何使用各種的工具來檢查及維護檔案系統；此外，他還應該知道存取像是在軟碟或光碟片上的各式檔案系統，所需要用到的各種命令與檔案。

檔案系統類型

表 6-1 列舉了 Linux 核心版本 2.4.10 所支援的檔案系統類型。一直不斷都會有新型的檔案系統陸續被加到系統上，而且還有一些測試中的檔案系統也沒有列出來。如果想知道所用的核心支援哪些類型的檔案系統，可以查看*/proc/filesystems* 這個檔案；在建立核心的時候，還可以選擇所要支援的檔案系統類型。關於建立核心的議題，請參考第七章。

表 6-1 Linux：檔案系統類型

檔案系統	類型	說明
Second Extended filesystem	ext2	通用的 Linux 檔案系統
Reiser filesystem	reiserfs	日誌型（journaling）檔案系統
Third Extended filesystem	ext3	另一種日誌型（journaling）檔案系統可向後相容於 ext2
Minix filesystem	minix	原始 Minix 檔案系統，很少用
ROM filesystem	romfs	一種小型唯讀檔案系統，主要用於 ramdisk
CRAM filesystem	minix	壓縮的唯讀檔案系統通常用於 PDA
Network File System（NFS）	NFS	可經由網路存取遠端的檔案
UMSDOS filesystem	umsdos	把 Linux 安裝在 MS-DOS 分割區時所用
DOS-FAT filesystem	msdos	存取 MS-DOS 檔案
VFAT filesystem	vfat	存取 Windows 95/98 檔案
NT filesystem	ntfs	存取 Windows NT 檔案
HPFS filesystem	hpfs	OS/2 的檔案系統
/proc filesystem	proc	為 ps 提供行程資訊
Device filesystem	devfs	另一種顯示目錄 */dev* 中裝置檔案的方式
ISO 9660 filesystem	iso9660	大部分光碟片所用的格式

檔案系統	類型	說明 （續）
Joliet filesystem	iso9660	ISO 9660 的延伸，可以處理 unicode 檔名
UDF filesystem	udf	最現代的光碟檔案系統
System V filesystem	sysv	存取 System V 族系的檔案
Coherent filesystem	coherent	存取 Coherent 的檔案
UFS filesystem	ufs	存取來自 UFS 檔案系統的檔案，像是 SunOS、BSD 或 Tru64 Unix
BFS filesystem	bfs	存取 SCO Unixware 上的檔案
EFS filesystem	efs	存取在舊有 Irix 版本上的檔案
ADFS filesystem	adfs	存取 Acorn 分割區的檔案
AFFS filesystem	affs	允許從標準的 AmigaOS 檔案系統分割區存取檔案
Apple Mac filesystem	hfs	存取 Macintosh 的檔案
QNX4 filesystem	qnx4	存取 QNX4 分割區的檔案
JFFS filesystem	jffs	快閃記憶裝置之檔案系統
Novell filesystem	ncpfs	經由網路存取 Novell 伺服器的檔案
SMB filesystem	smbfs	經由網路存取 Windows 伺服器的檔案
Coda filesystem	coda	類似於 NFS 的先進網路檔案系統
RAM filesystem	ramfs	RAM 磁碟檔案系統
Temporary filesystem	tmpfs	另一個全部在 RAM 記憶體上的檔案系統

各種類型的檔案系統都有各自的屬性和限制；例如，MS-DOS 檔案系統限定檔名最多 8 個字元，副檔名最多 3 個字元，而且只能存取既有的 MS-DOS 磁片或分割區。在 Linux 系統中，大都會使用 Second Extended（*ext2*），這是特別為 Linux 所發展出來的檔案系統，可支援 256 個字元的檔名長度，與高達 32 terabyte 的檔案系統容量，以及其它許多好康的功能；或者也可以考慮用更新的 Third Extended（*ext3*）或 Reiser（*reiserfs*）檔案系統。早期的 Linux 所用的是 Minix 檔案系統，以及目前已廢棄不再支援的 Extended 檔案系統；當初所以會使用 Minix 檔案系統是可以理解的，因為 Linux 原本就是在 Minix 下編譯出來的，而且 Linus 本人對 Minix 檔案系統十分熟捻，所以就直接把這個功能做到核心中了。原本在舊版的 Linux 核心可以使用的 Xia 與 Xenix 檔案系統現在則已經不再支援了。

Second Extended 與 Reiser 檔案系統的最主要差異在於後者是日誌式的類型。日誌化是一項先進的功能，它可以追縱對檔案系統所做的任何變更，而且在像是系統當機或斷電等突發事件之後，也比較容易迅速將受損的檔案系統復原。Third Extended 為 Second Extended 檔案系統的後繼者，也是屬於日誌式的檔案系統，它的優點是可向後相容於 Second Extended 檔案系統，但由於才出現不久，所以還不像 Second Extended 與 Reiser 那樣普遍。

我們很少會需要用到 ROM 檔案系統，這是一個極為迷你的檔案系統，只允許讀取資料的動作，無法進行寫入的動作，通常只用在系統組態設定與啟動時的記憶體虛擬磁碟機（ramdisk），甚或用於 EPROMS。Cram 檔案系統也屬於這一類，同樣適用於唯讀記憶體（ROM），它可壓縮檔案內容，主要是應用在記憶空間極為珍貴稀少的嵌入式系統。

如果想要在既有的 MS-DOS 分割區上安裝 Linux，可以選用 UMSDOS 檔案系統。它可以將 Linux 安裝在現存 MS-DOS 分割區內的某個個人專屬目錄下，讓新手有機會玩玩 Linux 而不需重新分割硬碟，這是個不錯的方式，只是會犧牲系統效能；另一方面，DOS-FAT 檔案系統也可用來直接存取 MS-DOS 檔案；經由 VFAT 檔案系統則可存取 Windows 95 或 98 所建立的分割區；而使用 NTFS 可以存取 Windows NT 檔案系統的資料；若欲存取 OS/2 檔案系統，則需使用 HPFS 檔案系統。

DOS-FAT 檔案系統還有一個 CVF-FAT extension 的功能，就可以在 Linux 下直接存取經過 DoubleSpace/DriveSpace（Microsoft）或 Stacker（Stac）壓縮過的磁碟分割區。進一步的細節可參考 Linux 核心的資料 *Documentation/filesystems/fat_cvf.txt*。

/proc 是一個虛擬的檔案系統，也就是說，它並不佔實際的磁碟空間；可參閱第五章〈/proc 檔案系統〉一節【註】。

如同 */proc* 一般，*devfs* 也是一個虛擬的檔案系統，用來取代原本的 */dev* 目錄（參閱本章稍後〈裝置檔案〉一節之說明）。它的優點是系統管理者不需自己動手建立一些特殊檔案，而可以由核心視實際需求自動建立。

大多數的 CD-ROM 都使用 ISO 9660 檔案系統，早期也稱為 High Sierra 檔案系統，在其它的 UNIX 上則簡稱為 *hsfs*。如同 MS-DOS 檔案系統，它不但限制檔名長度，並且所儲存的檔案相關資訊也極其有限；但大多數的 CD-ROM 都有提供 Rock Ridge Extension 延伸功能來擴充 ISO 9660，這樣一來，核心的檔案系統驅動程式就可替每個檔案指定長檔名、檔案所有者和使用權限；其結果是，在 MS-DOS 作業系統下使用 ISO 9660 光碟，你只能看到 8.3 格式的檔名（主檔名不超過 8 個字元，而副檔名不超過 3 個字元），但是在 Linux 下則可以得到「真正」完整的檔案名稱。

此外，Linux 現在還支援 Microsoft Joliet extension（也是針對 ISO 9660 的擴充），能夠處理以 Unicode 編碼的長檔名，雖然目前尚未廣泛使用，但有可能前途看好，因為 Unicode 已逐漸被國際所接受，成為字元編碼的標準。

註　注意，在 Linux 上 */proc* 檔案系統的其格式並不同於 SVR4 版本的 UNIX（例如：Solaris 2.x）的格式。在 SVR4，每個正在執行中的行程在 */proc* 都有一個單獨的“檔案”項目，可以用特定的 *ioctl()* 系統呼叫來開啟與處理，以取得該行程的資訊；相反地，Linux 每個行程在 */proc* 會有一個對應的子目錄，內含好幾個關於該行程資訊的檔案，可透過一般的 *read()* 與 *write()* 系統呼叫取得。

最近 Linux 又增加了對 UDF 的支援，這是設計來給 CD-RW 及 DVD 使用的一種新的檔案系統。

其次是六種在其它平台上很普遍的檔案系統類型，分別為 UFS、EFS、BGS、System V 和 Coherent — 最後兩項其實是由相同的核心驅動程式所控制，只不過所使用的參數有些微的差異。Linux 支援這些類型的檔案系統，以達到雙重開機的功能及與其它平台交流的目的；在其它的作業系統中，若有使用以上述其中任一格式所建立的檔案系統，都可以從 Linux 來存取那些檔案。

最後，還有許多檔案系統可用來存取一些非 DOS 或 UNIX 家族的作業系統所建立的分割區上的資料。這類檔案系統的支援包括 Acron Disk Filing System（ADFS）、Amiga Fast File System（除了在 Amigas 機器上之外，並不支援軟碟）、Apple Mac HFS 與 QNX4 檔案系統。這些特殊的檔案系統大多只能用於特定的硬體架構；例如，你不會在 Intel CPU 的個人電腦上使用 Amiga HFS 檔案系統來格式化硬碟。如果需要使用這些檔案系統的驅動程式，可參閱隨附於軟體的相關資訊說明；但有些還只是在實驗階段。

除上述可用來存取本地硬碟的檔案系統外，還有一些檔案系統可用來存取遠端資料，不過這個議題並不在本書的討論範圍內，可自行參閱 Olaf Kirch 與 Terry Dawson 合著（O'Reilly 公司所出版）的《Linux Network Administrator's Guide》一書。

掛載檔案系統

要在 Linux 下使用任何檔案系統，必須先將其「掛載」（mount）於某個目錄之下，這會讓該檔案系統上的檔案，看來似乎就是存在於指定的目錄下，讓你可以存取它們。

在說明如何掛載檔案系統之前，必需先提到，有些發行套件會有自動掛載的設定，只要將磁碟或 CD 片置入軟碟機或光碟機，就可以如同在其它平台一般地使用這些儲存媒介。但對每位使用者而言，有時總會遇上需要自己動手直接掛載（mount）或卸載（unmount）儲存媒介的情形，至於要如何設定自動掛載，稍後將會告訴你。

`mount` 命令可以達成掛載的功能，但通常必須以 *root* 的身分來執行這個命令；後文將會提到，只要該裝置有被列在 */etc/fstab* 檔案中，一般使用者也可以使用 `mount` 命令。`mount` 的格式如下：

```
mount -t type device mount-point
```

type 就是檔案系統的類型（如表 6-15 中所列），*device* 是檔案系統所在的實際裝置（就是 */dev* 下的裝置檔），*mount-point*（掛載點）則是要掛載該檔案系統的目錄名稱。使用這個命令之前，必須事先建立該目錄。

例如，假使在 */dev/hda2* 分割區上有 *ext2* 檔案系統，而想要把它掛在 */mnt* 這個目錄下，則可以用下列命令：

```
mount -t ext2 /dev/hda2 /mnt
```

一切順利的話，這樣應該就可以使用 /mnt 下的檔案系統了。同樣地，若要掛上一個在 Windows 作業系統下製作的 DOS 格式軟碟，可使用以下命令：

```
mount -t msdos /dev/fd0 /mnt
```

這麼一來，就可以透過 */mnt* 使用 MS-DOS 格式磁片上的檔案。注意，*msdos* 表示用的是舊的 DOS 格式，所以檔名的限制為 8 個字元的主檔名再加上 3 個字元的副檔名。若改成 *vfat*，則是使用 Windows 95 所引進的新型格式；當然，寫入軟碟或硬碟時，也必需依照這種格式。

mount 還有很多選項，可以用 -o 來指定；例如，想讓 MS-DOS 和 ISO 9660 檔案系統具有「自動轉換」的能力 — 也就是把斷行符號從 MS-DOS 的 CR-LF 格式轉成 UNIX 慣用的僅有 newline（LF）的格式 — 就像這樣：

```
mount -o conv=auto -t msdos /dev/fd0 /mnt
```

使用這個命令會為那些附檔名不屬於二位元類型（諸如：*.bin* 或 *.exe* 等）的檔案開啟自動轉換的功能。

另一個 mount 常用的選項是 -o ro（和 -r 相等），它會將檔案系統掛載成唯讀（read-only）狀態，任何對該檔案系統寫入的動作都會導致“permission denied”的錯誤訊息。對於光碟之類不能寫入的媒介，有必要把檔案系統掛為唯讀狀態；不加 -r 的參數，雖然也可以將 CD-ROM 掛載成功，但會收到像下面這樣惱人的警告訊息：

```
mount: block device /dev/cdrom is write-protected, mounting read-only
```

所以，應該使用如下的命令：

```
mount -t iso9660 -r /dev/cdrom/mnt
```

當要嘗試掛上一片防寫的磁片時，也同樣需要這麼做。

mount 的手冊頁面會列舉出所有的選項；雖然你也許不會對所有的選項立刻都感到興趣，但是其中的某部分有一天或許會用到。mount 還有另一種使用方式，即 mount -a，這會將 */etc/stab* 中凡未標上 noauto 參數的檔案系統全數掛載上來。

相對於「掛載」的動作就是「卸載」。卸載檔案系統會產生兩個作用：將系統緩衝區內所暫存的資料寫回該磁碟上檔案系統，使其資料與記憶體同步，然後讓檔案系統與原先的掛載點脫離，之後就可以將其它的檔案系統再掛到該掛載點上。

若要卸載檔案系統，必需使用 umount（注意，這個命令比 unmount 少了第一個 n）：

```
umount /dev/fd0
```

這會卸除裝置 */dev/fd0* 上的檔案系統；同樣地，要卸除目前掛載於某一目錄上之任何檔案系統，可以使用如下的命令：

```
umount /mnt
```

注意，對於可抽換的儲存媒介（如軟碟、光碟等），在尚未卸載之前，不應將其取出或更換，因為這會造成系統對於該裝置的相關資訊與實際狀況不一致，並引發一堆問題。無論何時，當你想更換磁片或光碟片，應先使用 umount 命令將其卸載，換好磁片之後再重新掛載。當然，若是使用唯讀的 CD-ROM 或防寫磁片，就裝置本身是不至於發生不同步的情況，但還是可能會遇上其它的問題，例如：有些光碟機若不先行卸載，是無法退片的。

對軟碟磁片上的檔案系統進行讀寫時，就像對硬碟讀寫的情形一樣，會先被暫存在記憶體中，這表示在你將資料讀出或寫入軟碟時，磁碟機可能沒有立即的動作，系統會以非同步方式來處理軟碟的輸入／輸出，只有在絕對必要的時候才會進行真正的讀寫動作；因此，如果將一個小檔案複製到磁片，但磁碟機燈號並未亮起，這時也不用驚慌，這些資料終究還是會寫到磁片上；你可以用 sync 這個命令來觸發暫存資料的實際寫入動作，也就是強制系統將檔案系統的暫存資料全部寫入磁碟中；將檔案系統卸載時也會引發這個動作。

如果要讓一般使用者能夠自行掛載或卸載特定的裝置，可以有兩種作法：一是在 */etc/fstab*（本節稍後會提到）檔案中為該裝置加上 user 選項，這使得所有使用者都可以用 mount 和 umount 命令來掛載或卸載該裝置；另一種方法就是使用 mount 的前端界面（front-end）程式，這些程式執行時會 setuid 為 *root*，此時使用者識別碼 uid 暫時被設成 *root*，而讓一般的使用者也能擁有 *root* 的權限來掛載特定裝置。一般來說，你並不會想讓一般使用者任意掛載或卸載硬碟分割區，但對於光碟機與軟碟機的使用則可以放寬一點。

嘗試掛載檔案系統時可能會發生一些錯誤，不幸的是，對於許多不同的問題，mount 所回應的卻是千篇一律的錯誤訊息：

```
mount: wrong fs type, /dev/cdrom already mounted, /mnt busy, or other error
```

最簡單的就是 wrong fs type：這個訊息表示可能在 mount 時指定了錯誤的檔案系統類型；如果沒有指定任何類型，Linux 會從 superblock 去猜測所使用的檔案系統（通常只有對 *minix*、*ext2* 與 *iso9660* 才行得通）；如果還是不行，mount 會去嘗試核心所支援的所有檔案系統（列於 */proc/filesystems*）；如果再不成，mount 動作就會失敗。*device* already mounted 則代表該裝置已經掛到了其它的目錄上了。mount 命令若不加任何參數，可以找出那些裝置已經掛上了，以及其掛載點的位置：

```
rutabaga# mount
/dev/hda2 on / type ext2 (rw)
/dev/hda3 on /windows type vfat (rw)
/dev/cdrom on /cdrom type iso9660 (ro)
/proc on /proc type proc (rw,none)
```

這裡可以看到有兩個硬碟分割區，其中一個是 *ext2* 類型，另一個是 *vfat*，光碟則是掛在 */cdrom* 之下，還有一個 */proc* 檔案系統。每行的最後一個欄位列出了該檔案系統掛載時的參數選項（例如：`rw`），等一下還會再做說明。需注意到光碟通常會掛在 */cdrom*，如果經常使用光碟，那麼建立一個像是 */cdrom* 這樣特別的目錄，將該裝置掛在該目錄下會比較方便；*/mnt* 目錄則經常是用來掛載一些像是磁片之類的暫時性檔案系統。

mount-point `busy` 是個相當奇怪的警告訊息，基本上這表示在掛載點（mount-point）之下正在進行某些事情，使得你無法將檔案系統掛上，通常是在該目錄以下可能有某個檔案正在使用中，或是某個行程目前的工作目錄就在掛載點以下的某一點。在使用 `mount` 命令的時候，務必確定 root shell（也就是目前用的 shell，此刻你應該是 *root* 的身分）目前的工作目錄不是在掛載點的下面，你可以用 `cd / ` 命令，強迫 shell 回到根目錄，來確保這一點。當然，不可以有兩個檔案系統使用相同的掛載點（*mount-point*）。你可直接用 `mount` 命令不加任何參數來列出所有已被掛載的檔案系統，檢查看是不是已經有另外的檔案系統掛載到同一點上。

如果你收到 “other error” 這樣的錯誤訊息，八成覺得很無奈，因為實在是沒多大幫助。有幾種可能的錯誤原因：如果這個檔案系統有資料或儲媒的錯誤，`mount` 可能會告訴你，它無法讀取檔案系統的超級區塊（superblock）— 在 UNIX 類型的檔案系統下，所謂的 superblock 是檔案系統的一部分，其中儲存了與整個檔案系統有關的檔案資訊及屬性資訊。還有，假如嘗試掛上光碟機或磁碟機，但是並沒有光碟或磁片放在裡面，則會收到下面這樣的錯誤訊息：

```
mount: /dev/cdrom is not a valid block device
```

軟碟磁片特別容易故障（可能遠出乎你所意料之外），光碟則容易覆蓋灰塵、受刮傷、印上指紋，以及正反面放置錯誤 — 如果你試著把加拿大民歌手史丹．羅傑的 CD 掛為 ISO 9660 的格式，也很可能會遭遇類似的問題。

此外，記得確定所用的掛載點確實已經存在（例如：/mnt），倘若不存在，只要用 mkdir 命令就可以建立該目錄。

如果在掛載或使用某個檔案系統時發生問題，也可能是檔案系統上的資料損壞了，有幾種工具可以修理某些類型的 Linux 檔案系統，請參考稍後的〈檢查與修復檔案系統〉一節。

在系統的啟動階段，會自動掛載幾個檔案系統，這是由 */etc/fstab* 檔案所負責設定的，每一個在啟動時應該被掛載上來的檔案系統，都會在 */etc/fstab* 裡有一行對應的紀錄，其格式大致如下：

```
device mount-point type options
```

在此，*device*、*mount-point* 和 *type* 都與 mount 命令中所用的意義相同，*options* 是一組由逗點分隔的選項（也就是用在 mount -o 之後的選項）。

一個簡單的 */etc/fstab* 檔就如下所示：

```
# device      directory     type         options
/dev/hda2     /             ext2         defaults
/dev/hda3     /windows      vfat         defaults
/dev/cdrom    /cdrom        iso9660      ro
/proc         /proc         proc         none

/dev/hda1     none          swap         sw
```

這個檔案的最後一行指定了一個置換分割區（swap partition），關於這點於後文〈管理置換空間〉一節會再討論。

mount(8) 手冊頁面列舉了 *options* 所有可能的值，如果想要指定兩個以上的參數，可以用逗號（中間不加空白）來分隔，例如：

```
/dev/cdrom       /cdrom         iso9660   ro,user
```

user 選項允許 *root* 之外的使用者去掛載檔案系統，如果選擇這個選項，使用者就可以用類似下面的命令把裝置掛上：

```
mount /cdrom
```

注意，如果你在 mount 時只指定了裝置或掛載點（而非兩者同時指定），系統將會到 */etc/fstab* 中查尋該裝置或掛載點，並且使用該檔案中記錄所指定的參數；因此當手動掛載在 */etc/fstab* 中有列出的裝置時，這樣就會輕鬆多了。

大多數的檔案系統會使用 defaults 選項，這會啟用許多個預設的掛載選項，例如：rw（可讀寫）、async（以非同步方式將檔案系統的 I/O 資料暫存到記憶體中，可提升系統的效率）... 等等，除非你有特定的需求必需變更這些預設參數，否則大部分檔案系統都是以 defaults 作為參數，而唯讀的裝置像是 CD-ROM 就用 ro 選項。還有一個可能會很有用的參數 umask，可對權限位元設定預設遮罩，這對於一些外來的檔案系統特別有用。

`mount -a` 命令會把列在 */etc/fstab* 內的所有檔案系統都掛上。通常系統在啟動時，在 */etc/rc.d* 目錄（或其它發行套件用來存於組態設定檔的目錄位置）中的某一個 script 檔（例如 *rc.sysinit*）會來執行 `mount -a` 命令；這樣一來，所有列在 */etc/fstab* 中的檔案系統在系統啟動後就可以使用了：包括硬碟分割區、光碟機... 等等，都會被自動掛上。

root 檔案系統是一個特例；它是掛在 / 下，它通常包含 */etc/fstab* 和 */etc/rc.d* 下的 script 檔。為了讓這些檔案馬上可以使用，核心必須在啟動時直接掛載 root 檔案系統，因此，root 檔案系統的裝置必須被編寫進核心映像檔（kernel image）之內，如果要改變 root 檔案系統所用的裝置，必須另外借助 `rdev` 命令〔請參考第五章的〈使用開機片〉一節〕。當系統啟動時，核心會連續嘗試數種檔案系統類型以便將 root 檔案系統掛上。倘若在啟動時，核心印出如下的錯誤訊息：

```
VFS: Unable to mount root fs
```

這表示可能發生了下列幾種情形之一：

- 編寫入核心映像檔的 root 裝置不正確。
- 在編譯核心時，沒有把適用 root 裝置的檔案系統類型編譯進去（關於細節部分，請參考第七章的〈建立核心〉一節）。
- root 裝置受到某種破壞。

前述的任何一種狀況，都會使核心無法繼續進行，並且會不知所措（panic）；至於解決的方式，請參考第八章的〈危機處理〉一節。如果問題是出在檔案系統損壞，通常都能夠修復；可參閱本章稍後〈檢查與修復檔案系統〉一節的說明。

要掛載某個檔案系統並不一定得將它列在 */etc/fstab* 中，但若要以 `mount -a`「自動」掛載檔案系統，或是讓一般的使用者也能夠掛載檔案系統（`user` 選項），就必須要將其列於 */etc/fstab* 中。

自動掛載

如果經常需要存取許多不同的檔案系統，特別是那些透過網路存取的檔案系統，你可能會對於 automounter 這個特殊的 Linux 核心功能感到很有興趣。它結合了核心功能、伺服程式（daemon），以及一些設定檔，讓系統可自動偵測使用者對某項檔案系統的要求，並自動掛載檔案系統，而完全不需要驚動到使用者。當檔案系統閑置一段時間未使用時，automounter 會自動卸載它，以節省記憶體資源，或紓解網路的流量。

若要用 automounter，必須先在建立核心的時候打開自動掛載（*autofs*）這項功能（可參閱第七章〈建立核心〉一節），而且必須同時開啟 NFS 選項。

然後還要啟動 automount 這個 daemon。由於 *autofs* 這項功能還很新，所以不一定每個 Linux 發行套件版本都有涵蓋，可以先找一下 */usr/lib/autofs* 目錄，若這個目錄不存在，就要想辦法取得 *autofs* 套件，再依照指示來編譯與安裝。

注意，automount 功能的支援有兩個版本：即第 3 版與第 4 版，大部分發行套件所包含的還是第 3 版，所以下面就用它為例子來做說明。

檔案系統可自動掛載在任何位置，但為了說明簡便起見，假設所有要自動掛載的檔案系統都是在 */automount* 這個目錄以下；如果自動掛載點遍佈系統各處，那就需要啟動多個 automount daemon，讓它們各自負責一個目錄。

如果是自行編譯 *autofs* 套件的話，則可用 *sample* 目錄下的組態設定檔範例作為基礎，將它們修改符合自己的需求；方法很簡單：首先把 *sample/auto.master* 與 *sample/auto.misc* 這兩個檔案複製到 */etc* 目錄下，再把 *sample/rc.autofs* 複製到系統存放開機 script 的目錄（在此假設是 */etc/init.d*）下，並命名為 *autofs*。

第一個要編輯的設定檔是 */etc/auto.master*，它需列出 automounter 用來掛載磁碟分割區的所有目錄，也就是所謂的掛載點（mount point）；在本範例中只用了一個分割區，所以在此只需加入一行類似以下的記錄就好了：

```
/automount  /etc/auto.misc
```

這個檔案的每行包含了兩個項目，中間用空白隔開。第一個項目用來指定掛載點（mount point）。第二個項目是所謂的對映檔案（map file），用來指定要自動掛載的裝置或分割區所在位置，以及掛載的方式；而每個掛載點都要有一個 map file。

在本例中，*/etc/auto.misc* 的內容看起來就像這樣：

```
cd        -fstype=iso9660,ro      :/dev/scd0
floppy    -fstype=auto            :/dev/fd0
```

其中每一行指定了一個要自動掛載的特殊裝置或分割區，每行的格式都包含兩個必要的欄位與一個選擇性的欄位，中間用空白隔開。第一個欄位是必要的，用來指定此裝置或分割區自動掛載的地方；這個欄位會被附加到前面所說的掛載點之後，因此 CD-ROM 會自動掛載到 */automount/cd*。

第二欄（可有可無）則設定掛載時要使用的選項，相當於在你自己手動掛載時，指定給 mount 的選項，但是要把原先的 -t 改成 -fstype=。

第三個欄位則是指定所要掛載的裝置與分割區，以本例來說，我們分別要掛載第一台 SCSI CD-ROM 光碟機與第一台軟碟機。注意，此欄位之前的冒號（:）是用來隔開主機名稱與裝置檔目錄，不可以省略。第三欄完整的格式是 sourcemaster:/source（用法與 mount 命令相同），因為這些裝置都是在自己的電腦上，所以 sourcemaster 部分就不見了。若被掛載的來源是在 NFS 伺服器上，就必須在設定檔中使用下列完整的格式：

```
sources    -fstype=nfs,soft    sourcemaster:/sources
```

當你完成了組態設定檔的編輯工作，讓它能反應系統的實際需求時，可以輸入下列命令來啟動自動掛載的 daemon（請以系統上實際存放 script 的路徑取代下面例子中的路徑）：

```
tigger# /etc/init.d/autofs start
```

因為這個命令相當沈默，可能會讓你搞不清楚 automounter 到底啟動了沒有，如果想進一步確認，有個方法可以知道：

```
tigger# /etc/init.d/autofs status
```

光這樣還是很難從它的輸出中搞清楚 automounter 是否真的在運作，所以最好的選擇，就是看看這個 automount daemon 行程是否存在：

```
tigger# ps aux | grep automount
```

如果這命令顯示了 automount 行程，就沒什麼問題了；否則的話，就必須再檢查一下組態設定檔；也有可能是因為核心根本沒有支援相關功能，比如沒有將支援 automount 的功能編譯進核心來；或是把它編譯成模組的形式，但卻未將模組安裝到系統。如果是後者，可以用下列命令來修正這個問題：

```
tigger# modprobe autofs
```

如果還是不行，則需改用以下命令【註】：

```
tigger# modprobe autofs4
```

如果這樣就搞定了，你可能會想要把 modprobe 與 autofs 等命令都放到系統的啟動設定檔上，像是 */etc/rc.d/rc.local*、*/etc/init.d/boot.local*（視所用之 Linux 發行套件而定）。

若是一切順利，只要對掛載點下的目錄做些存取動作，automounter 就會自動把適當對應的裝置或分割區掛載到系統上。例如輸入下列命令：

```
tigger$ ls /automount/cd
```

automounter 就會自動掛載 CD-ROM，而 ls 即可列出光碟的內容，與一般事先掛載方式唯一的差異，是在命令動作之前會有一點遲緩的現象。

為了節省系統資源，automounter 在經過一段固定時間之後（預設是五分鐘），若發現你沒有對自動載入的裝置進行任何動作，就會自動進行卸載。

註　在下一章將會討論到 modprobe。

此外，automounter 還提供了一些先進的功能，舉例來說，除了從檔案讀取對映表之外，它也可以存取系統資料庫，甚或執行一個程式而利用其輸出來作為對映資料，詳細情形可參考 autofs 與 automount 的手冊頁面。

建立檔案系統

mkfs 命令可以用來建立檔案系統。建立一個檔案系統，就類似於「格式化」（format）一個分割區或軟碟，以便於儲存檔案。

每種檔案系統類型都有其對應的 mkfs 命令。例如，MS-DOS 檔案系統可以用 mkfs.msdos 命令來建立，*ext2* 檔案系統可以用 mkfs.ext2 來建立 ... 等。而 mkfs 本身則是個前端界面程式，它會執行適當的 mkfs.*<fs-type>* 命令來建立各種類型的檔案系統【註】。

在安裝 Linux 時，可能已經自己動手用 mke2fs 這類工具來建立檔案系統（也可能是安裝軟體代勞的）；事實上，mke2fs 與 mkfs.ext2 是一樣的，它們的程式相同（在很多系統中，這兩者其中之一是另一個的符號連結），但是後者 mkfs.*<fs-type>* 這樣的命名法比較方便，容易讓 mkfs 根據不同的檔案類型來呼叫特定的程式以建立適當的檔案系統。如果你沒有 mkfs 這個前端界面，則不妨試著直接用 mke2fs 或 mkfs.ext2。

假定你使用 mkfs 這個前端界面，可以下面的命令來建立檔案系統：

```
mkfs -t type device blocks
```

type 就是所要建立的檔案系統類型（如表 6-1）；*device* 是要在上面建立檔案系統的裝置名稱（例如，軟碟可能是 */dev/fd0*）；*blocks* 是所要建立的檔案系統大小，以區塊為單位，這個參數可有可無，但有些類型的檔案系統不會自動偵測裝置的大小，或是無法確定時，就必須輸入 *blocks*。

舉例來說，要在軟碟片上建立一個 *ext2* 檔案系統，可以使用下面的命令：

```
mkfs -t ext2 /dev/fd0
```

若改成 -t msdos 則可以建立 MS-DOS 格式的磁片。

如同前節所描述的，現在可以將這塊磁片掛上，再把檔案拷貝過去，等等；記得在把它從磁碟機取出之前，要先行卸載。

註　最早期的 mkfs 本身是用來建立 Minix 檔案系統；在較新的 Linux 作業系統中 mkfs 就變則成了建立各種類型檔案系統的前端界面，而現在 Minix 檔案系統實際上是用 mkfs.minix 命令來建立的。

建立檔案系統時，會將所對應的實際裝置上的資料全部刪除（無論是磁片或是硬碟分割區），mkfs 在建立檔案系統之前通常不會給你任何提示，所以務必要搞清楚自己在做什麼。

在硬碟分割區上建立檔案系統，除了將適當的分割區名稱（例如 */dev/hda2*）當做 ***device*** 的參數輸入，其餘步驟與前面軟碟的例子完全相同。記得只能對硬碟的分割區建立檔案系統，不要嘗試在代表整台硬碟的裝置（像是 */dev/hda*）建立檔案系統；可以先使用 fdisk 來建立分割區，請參閱第三章〈建立 Linux 分割區〉一節。

在硬碟分割區建立檔案系統的時候，應該格外小心，一定要確定 ***device*** 和 ***blocks*** 這兩個參數是正確的，如果輸入錯誤的 ***device***，可能會破壞目前檔案系統中的所有資料；而如果指定了錯誤的 ***blocks***，可能會蓋掉其它分割區上的資料，所以務必確定 ***blocks*** 與 Linux 的 fdisk（-l 選項）所回報的分割區大小吻合。

在軟碟片上建立檔案系統時，最好先做低階格式化，這會把磁區（sector）與磁軌（track）等相關訊息寫入磁片，如此一來，*/dev/fd0* 或 */dev/fd1* 就可以自動偵測磁片的容量，當以 mkfs 建立檔案系統時，***blocks*** 參數即可省略。低階格式化的方法很多，可以用 MS-DOS 的 FORMAT 命令，或是用 Linux 的 fdformat 程式【註】。例如，要將第一台軟碟機中的磁片格式化，可用下面的命令：

```
rutabaga# fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
```

如果使用 fdformat 的 -n 選項，就會略過複查（verify）的步驟。

每一種檔案系統特定版本的 mkfs 皆有支援一些選項，對你可能很有幫助。大多數類型的檔案系統都支援 -c 選項，可在建立檔案系統的同時一併檢查儲媒上是否有壞掉的磁區，如果找到了損毀的磁區，它會將其標示起來，將來所有對該檔案系統的寫入動作會避開這些已損壞的磁區。這些選項會因檔案系統類型而異，所以必須放在 mkfs 命令的 -t *type* 參數之後，例如：

```
mkfs -t type -c device blocks
```

至於哪些檔案系統提供了哪些選項，可參考特定檔案系統版本的 mkfs 手冊頁面（例如，對 *ext2* 檔案系統而言，可以參考 mke2fs）。

註　若你所用的 Linux 套件是 Debian，應使用 superformat。

你可能並未安裝所有版本的 mkfs，在此情形下，當要建立檔案系統而系統上卻找不到有對應的 *mkfs.type*，那麼 mkfs 就會失敗。Linux 所支援的檔案系統類型大部分都可以在網路上某個地方找得到對應的 *mkfs.type*。

如果在執行 mkfs 時遭遇到困難，有可能是 Linux 在對實際硬體裝置進行讀寫時發生問題；對磁片來說，也許只是磁片壞掉了，但對硬碟來說，問題可能就嚴重了！例如，核心中的磁碟機驅動程式可能無法讀取硬碟，這有可能是硬體壞了，或者只是硬碟的實體規格參數設定錯誤。可參考各種版本之 mkfs 手冊頁面，並閱讀第三章有關於解決安裝問題的部分，道理都是一樣的【註】。

檢查與修復檔案系統

有時候會非常有必要檢查 Linux 檔案系統的一致性，並且在發現有錯誤或資料遺失時予以修復，這類錯誤通常都是在系統突然當機，或電力中斷時造成的，它會使核心無法將檔案系統的快取緩衝區與磁碟的實際內容同步。在大部分的情形下，這種錯誤還不至於很嚴重；然而，如果正在寫入一個大型檔案時突然當機，該檔案可能會遺失；而那些已被寫入的區塊則會被標記為「被佔用」（in use），但實際上卻不屬於任何檔案。還有一些情況，會意外地將資料直接寫入磁碟機（像是：*/dev/hda*）或分割區而導致錯誤發生。

fsck（FileSystem CheckK）命令可用來檢查檔案系統，並矯正任何發現的問題。類似於 mkfs，fsck 本身也只是各種特定類型檔案系統的檢查程式 *fsck.type* 其前端界面而已。類似 *mkfs.ext2* 一樣，fsck.ext2 是針對 *ext2* 檔案系統的檢查程式，而實際上它是 e2fsck 的符號連結。即使你沒有裝 fsck 前端界面，直接執行 e2fsck 也可以。

fsck 的用法很簡單，格式如下：

```
fsck -t type device
```

type 是所要檢查或修復之檔案系統類型（表 6-1），*device* 是指該檔案系統所存在的裝置名稱（硬碟分割區或軟碟機）。

例如，要檢查一個位在 */dev/hda2* 上的 *ext2* 檔案系統，可以使用下列的命令：

```
rutabaga# fsck -t ext2 /dev/hda2
Parallelizing fsck version 1.06 (7-Oct-96)
e2fsck 1.06, 7-Oct-96 for EXT2 FS 0.5b, 95/08/09
/dev/hda2 is mounted. Do you really want to continue (y/n)? y
```

註　如果要建立一個可以給 CD-ROM 使用的 ISO 9660 檔案系統，它的程序可就不只是格式化檔案系統與複製檔案這麼簡單，詳細情形請參閱 CD-Writing HOWTO 文件之說明。

```
/dev/hda2 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts.
Pass 5: Checking group summary information.

Free blocks count wrong for group 3 (3331, counted=3396). FIXED
Free blocks count wrong for group 4 (1983, counted=2597). FIXED
Free blocks count wrong (29643, counted=30341). FIXED
Inode bitmap differences: -8280. FIXED
Free inodes count wrong for group #4 (1405, counted=1406). FIXED
Free inodes count wrong (34522, counted=34523). FIXED

/dev/hda2: ***** FILE SYSTEM WAS MODIFIED *****
/dev/hda2: ***** REBOOT LINUX *****
/dev/hda2: 13285/47808 files, 160875/191216 blocks
```

首先，對於已掛載的檔案系統，系統會要求確認是否要繼續進行，而且在這種情況下，如果 `fsck` 發現了任何錯誤並做了更正，你就必須重新啟動系統；這是因為 `fcsk` 對於儲存媒體所做的改變，可能無法立即反應回到系統的內部，來對檔案系統的狀態作同步之更新。通常最好不要在一個檔案系統還在掛載的狀態下來對它進行檢查。

如同在上面範例中所看到的，`fsck` 找到並更正了幾個問題，但由於這個檔案系統已經掛載到系統上，所以系統通知我們應該要重新啟動機器。

要如何在不掛載情況下來檢查檔案系統呢？除了 root 這個特例之外，你只要在執行 `fsck` 之前下 `umount` 命令就可以了，但 root 檔案系統在系統執行時是不能被卸載的。要檢查未掛載的 root 檔案系統，一個方法是用開機片加上 root 磁片的組合，例如發行套件所附的安裝磁片組。如此一來，root 檔案系統就含於磁片上，而硬碟中的 root 檔案系統則可保持未掛載的狀態，這樣就可以檢查硬碟上的 root 檔案系統了。細節可參閱第八章〈危機處理〉一節的說明。

另外還有一個檢查 root 檔案系統的方法，就是把它掛載為唯讀狀態 — 只要在 LILO 開機提示符號下使用 `ro` 選項就行了（可參閱第五章〈指定開機參數〉一節之說明）。然而，有些程式在開機時需要對 root 檔案系統進行寫入動作（例如：在開機時由 */etc/init* 所執行的程式），所以系統可能無法正常開機，或者是這些程式會執行失敗。正因如此，`ro` 選項通常還需要與另一個開機選項 `single` 搭配，也就是說，把系統啟動為單人模式，避免掉開機時額外的系統設定工作，就可以用 `fsck` 檢查 root 檔案系統，之後再以正常程序重新開機。

除了使用 `ro` 選項把 root 檔案系統掛載為唯讀狀態，也可以用 `rdev` 命令來設定核心裡面的 read-only 旗標。

很多 Linux 系統會把 fsck 放到 */etc/rc.d/rc.sysinit*，以便在開機時自動檢查各檔案系統，確認檔案系統沒有問題之後，才掛載檔案系統。一般的作法是系統先把 root 檔案系統掛載成唯讀狀態，並執行 fsck，然後執行下面的命令：

```
mount -w -o remount /
```

其中的 -o remount 會以新的參數重新掛載檔案系統；-w 選項（與 -o rw 相等）則是將檔案系統掛載為可讀可寫的狀態；因此在上面的例子中，root 檔案系統就被重新掛載為可讀寫的模式。

在開機時執行 fsck ，它會檢查所有的檔案系統，除 root 以外，都是在還沒掛上之前就先作檢查。fsck 完成後，系統會執行 mount 來掛上其它的檔案系統。檢查一下 */etc/rc.d* 目錄下的檔案，尤其是 *rc.sysinit*（如果在你的系統上有這個檔案的話），它會告訴你整個過程的進行方式。如果要關閉這個功能，只要在 */etc/rc.d* 下某個適當的啟動設定檔案中，把 fsck 相關的那行改成註解就行了。

有幾個選項可以用來傳給那些特定檔案類型的 fsck 命令；大多數的檔案系統都支援 -a 選項，它會自動確認所有 *fsck.type* 所提示的問題；-c 會檢查損壞的磁區；-v 會詳列操作過程中的訊息。如果使用前端界面，這些參數都應該加在 -t *type* 的後面來傳給 fsck。例如，下列的命令會使得 fsck 的輸出更詳盡：

```
fsck -t type -v device
```

參考 fsck 和 e2fsck 手冊頁面，可獲得更多的資訊。

並非所有 Linux 支援的檔案系統類型都有配套的 fsck；比如說，要檢查並修復 MS-DOS 檔案系統，就應該使用 MS-DOS 下的工具程式，像是：Norton Utilities，來完成這項工作。不過你至少一定可以找得到 Second Extended（ext2）、Minix 與 Xia 等幾個檔案系統的 fsck 程式。

第八章的〈危機處理〉一節，將會提供更多修復與檢查檔案系統的資訊。fsck 當然無法找出並修復系統上的所有錯誤，但卻可以處理大部分一般性的問題。如果你刪除了一個重要的檔案，目前並沒有簡單的復原方式 — fsck 不提供這項功能。有一個適用於 *ext2* 檔案系統的 undelete 工具也還在開發當中，所以還是請務必要保留備份，或是使用 rm -i，讓系統在刪除檔案之前，先與你再確認。

置換空間管理

置換空間（Swap space）泛指用來增加系統可使用記憶體容量的磁碟空間。在 Linux 下，置換空間是作為分頁（paging）之用：當實際記憶體不足時，就把部分記憶體頁（在 Intel x86 系統，一個分頁通常是 4096 bytes，至於其它架構的系統則不一定）暫時寫到磁碟上，必要時再讀回。分頁的運作程序相當複雜，但會針對某些特定的情形作最

佳化。Linux 的虛擬記憶體子系統允許幾個執行中的程式共享記憶體頁（memory page）。例如，當系統上同時有兩個 Emacs 在運作，記憶體中實際上只會有一份 Emacs 程式碼；此外，本文頁（text page）— 就是那些只含程式碼而不包含括資料的記憶頁 — 通常是唯讀的，所以在置換時可以直接從記憶體中釋放出來，而不必再寫到磁碟上；當有需要時，再從原來的可執行檔中讀出即可。

當然，置換空間不能完全取代實際的 RAM，因為磁碟存取的速度比 RAM 慢太多了，其差距有數個級數之多（1000 倍以上）。然而，在你需要同時執行多組程式，但是這些程式卻又無法一次全部放進實際的 RAM 中時，置換空間就會有所幫助；但如果在這些程式間快速切換，可能就會注意到，因為換頁的關係，隨著硬碟不斷讀寫置入置出，系統的反應會顯得有點遲滯。

Linux 支援兩種形式的置換空間：一種是獨立的磁碟分割磁區，另一種是在檔案系統中的置換檔案；系統最多可以有 8 個置換區，每個置換區的檔案或分割區最大可達 2 GB（這些上限值在非 Intel 的系統上可能會有所不同），算術高手們應該馬上就已算出系統最多可以有 16 GB 的置換空間（若真有人試著要使用這麼多的置換空間，不管他是否真是算術高手，作者倒是很樂於知道）。

注意，使用置換分割區的效果會比較好，因為磁碟區塊可以保證是連續的；倘若用置換檔案，磁碟區塊可能會分散在檔案系統的各處，這將會對效率有很大的影響。許多人只有在暫時需要增加額外的置換空間時，例如：當系統因缺乏實體記憶體與置換空間而不斷產生猛烈的置入置出現象，才會用置換檔。因應隨時而來的需求而增加置換空間，置換檔會是一個不錯的選擇。

幾乎所有的 Linux 系統都會使用某種類型的置換空間，通常是單一的置換分割區。第三章曾解釋如何在 Linux 安裝程序中為系統建立置換分割區；而在本節中，將描述增加或移除置換檔案與分割區的方法。假如系統已經設有置換空間，並且效能還算令人感到很滿意，那麼這一節的內容可能就不用太擔心。

系統擁有多少置換空間呢？`free` 命令會回報目前系統記憶體的使用情形：

```
rutabaga% free
         total               used       free     shared    buffers     cached
Mem:     127888            126744       1144      27640       1884      51988
-/+      buffers/cache:     72872      55016
Swap:    130748             23916     106832
```

所有的數字都是以 1024 byte 的區塊（block）為單位。在這個例子中，我們看到一個具有 127888 個區塊（大約是 127 MB）實際記憶體容量的系統，其中有 126744 個區塊（大約 126 MB）正在使用中；注意，系統實際上的記憶體數量應該比 *`total`* 那一欄列出的還要多，這個數字並不包含核心為了各種需要所用掉的部分。

shared 一欄列出了同時被好幾個行程所共享的記憶體數量，在這個例子中，我們看到大約 27 MB 的記憶體分頁是處於被共享的狀態，這表示系統記憶體的使用狀況還算良好；*buffers* 欄位顯示了核心的緩衝快取機制所使用的記憶體數量，快取緩衝區（前一節已簡要地提出說明）是藉由記憶體直接提供對磁碟讀取或寫入動作的服務，以加速磁碟的運作。隨著記憶體使用狀態的不同，緩衝區快取的容量會增加或減少，如果應用程式有需要記憶體時，系統會將它們回收。因此，儘管看到有 126 MB 記憶體正在使用中，但這些記憶體並非全都是由應用程式所佔用（雖說大部分還是）。*cached* 欄位是表示目前有多少記憶頁被核心快取（cached）住，以便稍後可以快速地存取。

因為作為緩衝區或快取的記憶體隨時可以回收給應用程式使用，我們將第一行中 *buffers* 與 *cached* 這兩個數字加起來，將其和拿去減掉已使用之記憶體總數 *used*，同時並加到尚未使用的記憶體 *free* 中，於是就得到了第二行（*-/+ buffers/cache*）中的兩個數字 — 分別表示目前實際上為應用程式所使用（*used* 欄位）與可為應用程式所用（*free* 欄位）的記憶體數量。

在第三行中（由 *Swap:* 開頭的那一行），我們看到置換空間的總數是 130748 個區塊（大約 128 MB）。在這個例子中，因為系統上還有大量的實體記憶體可用，所以只用掉了很少許的置換空間。如果你啟動了其它應用程式，就會有較多部分的快取緩衝區被用來放置這些程式，而置換空間通常是在系統無法藉由其它方式得到實際的記憶體時，才會使用的終極手段。

注意！*total* 所回報的置換空間容量，小於實際置換分割區與置換檔案容量的總和，因為各個置換區中會有幾個區塊是用來儲存每一個分頁在置換區中應如何對映使用的映射資訊，不過數量應該不多，大概每個置換區只有幾千個位元組！

如果你考慮要建立一個置換檔，可以先用 `df` 命令查看各個檔案系統所剩餘的空間，它會印出檔案系統的清單，並且顯示各檔案系統的容量以及目前使用的比例。

建立置換空間

要增設置換空間的第一步，就是要先建立一個檔案或分割區來當做新的置換空間。如果你想建立一個新增的置換分割區，可以使用 `fdisk` 來建立該分割區，其方法在第三章〈建立 Linux 分割區〉一節已說明過。

如果要建立置換檔，必須先開啟一個檔案，並塞入一連串的位元組資料進去，至於要塞多少垃圾，則看你所要的置換空間大小而定；簡單的方法是用 `dd` 命令。例如，要建立一個 32 MB 的置換檔，可以使用這個命令：

```
dd if=/dev/zero of=/swap bs=1024 count=32768
```

這會把 32768 個區塊（32 MB）的資料從 */dev/zero* 寫至 */swap* 中（*/dev/zero* 是一個特殊裝置，對它所做的任何讀取動作會一直傳回 0，其作用有點類似於反向的 */dev/null*）。在建立一個這麼大的檔案之後，最好用 sync 命令把檔案系統同步化，以免萬一系統當機而沒有將檔案完全寫到硬碟中。

一旦建立了置換檔或置換分割區之後，可以用 mkswap 命令來格式化置換區。如同第三章〈建立置換空間〉一節中所描述的，mkswap 命令的格式如下：

```
mkswap -c device size
```

其中，***device*** 是代表該置換分割區或檔案的名稱；***size*** 是以區塊（block）為單位來表示的置換區大小（再提醒一次，一個區塊等於 1024 bytes），通常可以不需要輸入這參數，因為 mkswap 自己會偵測分割區的大小；-c 選項也是可有可無，它可以在格式化置換區時順便檢查是否有損壞的磁區。

以剛才所建立的置換檔而言，可以用下面這行指來格式化：

```
mkswap -c /swap 32768
```

如果置換空間是分割區的型態，***device*** 那一欄必須輸入該分割區的名稱（例如：*/dev/hda3*），並將分割區的容量大小代入 ***size*** 欄位（也是以區塊為單位）。

若使用的是置換檔案（而非置換分割區），還需先將檔案的權限變更，如下所示：

```
chmod 0600 /swap
```

使用 mkswap 對置換檔格式化後，再執行 sync 命令以確保有關格式的資訊已經確實寫到新的置換檔上了。對於置換分割區而言，這倒是沒有必要。

啟用置換空間

為了讓新的置換空間生效，必須用 swapon 命令來啟用；例如，在建立了上述的置換檔並執行過 mkswap 與 sync 命令後，可以使用下面的命令：

```
swapon /swap
```

這會將新的置換區加到系統可用的置換空間中。可以再利用 free 命令來確定該區域是否真的已經啟用。

如果是新的置換分割區，則可以透過下面這個命令來啟用：

```
swapon /dev/hda3
```

和檔案系統一樣，你也可以將 swapon -a 命令加到諸如 */etc/rc.d/rc.sysinit* 的系統啟動檔中，以便在開機時自動啟用置換空間。這個命令會讀取 */etc/fstab* 檔（本章〈掛載檔案系統〉一節中已討論過），這個檔案中包含了檔案系統和置換區之相關訊息。swapon -a 命令會啟動所有在 */etc/fstab* 檔案中 ***options*** 欄位設為 sw 的項目。

因此，如果 */etc/fstab* 中包含下列項目：

```
# device      directory    type   options
/dev/hda3     none         swap   sw
/swap         none         swap   sw
```

在開機時，*/dev/hda3* 和 */swap* 這兩個置換區將會被啟用。對於每一個新的置換區，你應該為它在 */etc/fstab* 中新增一個項目。

停用置換空間

在一般情形下，要中止一項工作要比進行該項工作來得容易。你只需使用下面的命令就可以停用（disable）置換空間：

```
swapoff device
```

其中，*device* 是要關閉的置換分割區或置換檔名稱。例如，你想要關閉 */dev/hda3* 裝置下的置換功能的話，可以使用這個命令：

```
swapoff /dev/hda3
```

如果想停用一個置換檔，只需要在執行過 `swapoff` 命令後，再用 `rm` 把這個檔案移除。但請不要在停用一個置換檔之前就把它給移除，這會造成大災難。

使用 `swapoff` 來停用一個置換分割區之後，即可根據需要隨意重新利用該分割區：若要移除它，可以使用 `fdisk` 或個人所偏好的重分割工具程式。

此外，已停用的置換區在 */etc/fstab* 中對應的項目也要跟著移除，否則下次啟動系統時會有找不到置換區的錯誤訊息。

裝置檔

「裝置檔」（device file）的功用是讓應用程式可以經由核心來使用硬體裝置。就其本質而言，它並非真正的檔案，但從程式的觀點來說，它們看起來就和檔案一樣：你可以對它讀取、寫入、使用 *mmap()* 系統呼叫 ... 等等。在你使用到這麼一個裝置檔時，核心會辨識出 I/O 的請求（I/O Request），並將其傳給該裝置的驅動程式，以完成一些特定的 I/O 動作，例如從一個序列埠讀取資料，或是傳送資料到音效卡上。

裝置檔（雖然這名字取得並不是很好，但接下來仍將繼續沿用）提供了一種便利的方式，使得程式設計師不用瞭解裝置內部的運作方式就可以使用系統資源。Linux 就像大多數的 UNIX 一般，裝置的驅動程式本身就是核心的一部分。第七章的〈建立核心〉一節將告訴你如何建立自己的核心，並讓核心只包含系統上所具有硬體裝置的驅動程式。

幾乎所有的 UNIX 系統中，裝置檔都是放在 */dev* 目錄下，系統上的每個裝置在 */dev* 下都要有一個對應的項目；例如，*/dev/ttyS0* 對應到第一個序列埠，也就是 MS-DOS 下的 COM1；*/dev/hda2* 則對應到第一台 IDE 硬碟上的第二分割區。事實上，在 */dev* 下應該還有一些項目是系統所沒有的裝置，這些裝置檔通常是在安裝系統時所建立的，包括了所有可能裝置的項目；在你的系統上，這些裝置檔並不一定真得有實際的硬體裝置與之對應。

/dev 下還有一些虛擬的裝置，它們不會對應到任何的實體裝置。例如：*/dev/null* 就像是會吞噬資料的無底洞，任何對 */dev/null* 寫入的要求都會成功的執行，但是被寫入的資料都會消失無蹤；同樣地，先前已經展示過利用 */dev/zero* 建立置換檔的方法，任何對 */dev/zero* 讀取的要求都只能讀到一連串的 null byte。

使用 `ls -l` 列出 */dev* 下的裝置檔時，你會看到類似下面的輸出：

```
brw-rw----   1 root     disk        3,   0 May 19  1994 /dev/hda
```

/dev/hda 是對應到第一部 IDE 硬碟上的裝置檔；首先請注意權限欄的第一個字母是 `b`，表示這是一個「區塊裝置檔」（block device file）（回想一下，一般正常的檔案，其權限欄的第一個字元是 `-`、目錄是 `d` 等等），至於裝置檔則是用 `b` 來表示區塊裝置檔，或是用 `c` 來表示「字元裝置檔」（character device）。區塊裝置通常是像諸如硬碟這類的週邊裝置：對裝置的資料讀取與寫入都是以整個區塊為單位（區塊的大小取決於裝置硬體本身，並不一定是 Linux 下一般所指的 1024 byte 大小的 block），並且可以隨機存取該裝置；相對地，字元裝置通常只能循序地讀取與寫入，而其 I/O 動作可能都是以一個位元組（byte）為單位來完成的；序列埠就是字元裝置的一個例子。

請仔細觀察 `ls -l` 的輸出，其中的 *size* 欄位是兩個用逗號分隔的數字，第一個值代表「主要裝置編號」（major device number），第二個值代表「次要裝置編號」（minor device number）。當程式使用到裝置檔時，核心所收到的 I/O 請求是依據該裝置的主要與次要編號。主要編號通常是來指定了核心中的特定驅動程式，次要編號則是指定該驅動程式所管轄下的特定裝置。例如，所有的序列埠裝置的主要編號都是相同的，但次要編號則不同，核心依據主要編號將 I/O 的要求導向到適當的驅動程式上，而該驅動程式以次要編號來判斷要存取的特定裝置。在某些情形中，次要編號也用來指定某個裝置的特定功能。

/dev 下檔案命名沒有一定的慣例，說得不客氣一點，簡直是一團糟。因為核心本身並不在意 */dev* 中的檔名（核心只管主要與次要編號），Linux 發行套件的維護人員、應用程式設計師及驅動程式撰寫人員都可以自由選擇裝置檔的名稱。一般來說，編寫該裝置驅動程式的人會為該裝置建議一個名稱，但這些名稱後來會因為要遷就其它類似的裝置而被改變掉，在系統的發展過程中往往容易造成混淆與不一致性。希望你不會遇到這種問題，除非是較新的裝置驅動程式，特別是那些還在測試階段的，才比較有可能。

無論如何，發行套件中的裝置檔必需與核心版本與及所附的驅動程式都要能夠互相搭配。當你昇級核心或加入額外的驅動程式時（參閱第七章的〈建立新核心〉一節），可能還需要用 mknod 命令來增加新的裝置檔，此一命令的格式如下：

```
mknod -m permissions name type major minor
```

其中各個參數的意義如下：

- *name* 是所要建立之裝置的完整路徑，例如 */dev/rft0*。
- *type* 是裝置檔的種類，c 代表字元裝置，b 代表區塊裝置。
- *major* 是指該裝置的主要編號。
- *minor* 是指該裝置的次要編號。
- -m *permissions* 是一個可有可無的參數，這是將新裝置檔的使用權限位元設為 *permissions* 所指定的值。

例如，現在要新增某個裝置的驅動程式到核心，而其說明文件中告訴你必須建立一個區塊裝置 */dev/bogus*、主要編號為 42、次要編號為 0 的驅動程式，則可使用下面的命令：

```
mknod /dev/bogus b 42 0
```

若是利用附隨於多數發行套件中的 shell script */dev/MAKEDEV*，則建立新裝置的工作會更為簡易 — 你只要指定想要的裝置種類，*MAKEDEV* 會幫你找到它的主要編號與次要編號。

如果不指定 -m *permissions* 參數，新的裝置就會和一般新建立之檔案的預設權限相同，這個預設的權限，再經由目前之 umask 設定值來修正 — 通常是設為 0644。如果要將 */dev/bogus* 的使用權限設為 0666，可以使用：

```
mknod -m 666 /dev/bogus b 42 0
```

當然，你也可以用 chmod 在事後變更該裝置檔的使用權限。

為什麼裝置的使用權限如此重要呢？就像任何的檔案一樣，裝置檔的使用權限也決定了哪些人能夠使用該裝置，以及如何使用該裝置，在我們所看到的前一個範例中，*/dev/hda* 的裝置檔之使用權限為 0660，這表示只有檔案所有者以及所屬群組中的成員（這裡是指 *disk* 這個群組）能夠直接讀取與寫入該裝置（有關權限的部分在第四章的〈檔案所有者與使用權限〉一節中有詳細介紹）。

通常你不會想讓所有使用者直接對某個裝置進行讀寫，尤其是那些對應到磁碟機或分割磁區的裝置，否則任何人都可以在磁碟分割區上用 mkfs 來破壞系統上的資料。

對磁碟機和分割磁區而言，要破壞其資料至少要擁有對它們寫入的權限，然而可讀取的權限也是安全上的漏洞；如果給予某個使用者讀取磁碟分割區裝置檔之權限，他就能夠窺視其他使用者的檔案；同樣地，*/dev/mem* 這個裝置檔對應到系統的實際記憶體（通常是在進行某種極端偵錯手段時才會用到），如果你給予可讀取的權限，一些高竿的玩家們可能就可以窺看其他使用者登入時的密碼，這當然也包括 *root* 的密碼。

請確定系統上任何裝置的使用權限，應該恰如其分，與其正當的使用情形相符。像是序列埠、音效卡、虛擬主控台之類的裝置，可以開放給一般使用者而沒有安全上的顧慮，但是系統上其它絕大部分的裝置，應該限定只能讓 *root* 使用（以及那些在執行時會 setuid 為 *root* 的程式）。

某些在 */dev* 下的檔案，本身只是一個符號式連結（symbolic link），會連結到另外的裝置檔（通常是使用 `ln -s` 所建立的），這些連結可以讓使用者以一個比較普通的名字去使用某些裝置。例如，你有一個序列滑鼠，根據其所接的序列埠不同，這個滑鼠可能是經由 */dev/ttyS0*、*/dev/ttyS1*、*/dev/ttyS2* 或 */dev/ttyS3* 等裝置檔來使用，很多人會建立一個叫做 */dev/mouse* 的連結，並將其指向適當的序列裝置，如下：

```
ln -s /dev/ttyS2 /dev/mouse
```

以這種方式，就可以經由 */dev/mouse* 來使用滑鼠，而不必管滑鼠實際上到底是接到哪個序列埠。這個慣例也用在適用於 */dev/cdrom* 與 */dev/modem* 之類的裝置上 這些檔案通常都符號連結至 */dev* 中對應至 CD-ROM 與 modem 的實際裝置。

要移除裝置檔，只要像刪除一般檔案一樣，直接使用 `rm` 命令：

```
rm /dev/bogus
```

這並不會從記憶體或核心中把對應的驅動程式移除，而只是讓你無法再與該驅動程式相互溝通。同樣地，新增一個裝置檔，並不會在系統上加入該裝置的驅動程式；事實上，你可以為不存在的裝置驅動程式來新增裝置檔。核心之中若是有某一驅動程式，則裝置檔只不過是提供一個掛勾（hook）的界面功能，如此才可以讓外界使用該驅動程式。