第 二十一 章

系統管理 I——剖析檔案系統

之前曾經提過 UNIX 的檔案系統，就如同一個相當龐大的樹狀結構組織。但實際上，檔案系統(file systems)通常是許多目錄樹的大集合。在 UNIX 系統中，檔案系統通常是系統管理中，最需審慎處理的一環。本章中，將對此深入探究並了解其內部結構，再簡述如何建立磁碟分割區(partitions)及檔案系統，並掛上此分割區及如何檢驗其是否正常運轉。

檔案系統維護包括檔案、目錄及各裝置的資訊。系統管理員必須對其內部架構深入了解，方能確保檔案系統於正常狀態下運作。當檔案系統的資料不一致時，管理員需能在損失最少資料的情形下進行修復，以免導致日後更多資料損毀。再者當系統無法開機時，她更需要儘速地作好系統重灌與資料復原。假如她不能即時恢復系統運作，與復原回大部分資料時，將會被視為嚴重的失職。但您不要對此感到畏懼，UNIX 系統本身就提供了足夠的工具，讓您能照顧好整個檔案系統。

目　標

- 知悉兩種不同類型設備檔(device file)的定義與名稱。（21.1 與 21.2）
- 瞭解檔案如何存放於不同的磁碟分割區與檔案系統中。(21.4)
- 辨識檔案系統四個主要組件的功能。(21.5)
- 獲悉節點(inode)如何記錄檔案所使用的磁碟區塊位址(disk block addresses)。(21.5.3)
- 瞭解目錄於核心程式處理檔案時所扮演的角色。(21.6）
- 明白您在 UNIX 系統中會遭遇的各種檔案系統格式。(21.8)
- 使用 `fdisk` 與 `mkfs` 指令，在 Linux 系統中建立磁碟分割區與檔案系統。(21.9)
- 應用 `mount` 與 `umount` 指令，掛入(mount)與卸下(unmount)檔案系統。(21.10)
- 熟悉 `mount` 指令如何由 **/etc/fstab** 檔案獲得已掛入檔案系統的資訊。(21.10.3)
- 利用 `fsck` 指令檢查、修復檔案系統。(21.11)


Note

系統管理指令依各系統而異，功能與輸出也各有不同。有時，並不是所有的系統都提供同樣的指令。萬一當您的系統不能使用本章所提到的指令時，請您翻閱您的系統說明手冊；可能另有不同名稱但同樣功能的指令，或是該指令需加上不同的參數方能運作。

在本章中的指令，您大多數要以超級使用者（root）的身份才能使用。假如有必要的話，請您翻閱下一章中第 22.2 節部分，先行瞭解取得超級使用者權限會造成的後果與應注意事項。在本章中，您大部分需於 # 提示字元（代表您現在以root身份操作）下工作。

21.1 設備（The Device）

所有的設備，對 UNIX 系統而言都是以檔案來處理。啟動設備進行讀、寫後再關閉，就如同您處理一個檔案一般。對各設備的讀寫方式，其功能都已內建於系統核心(kernel)之中。所有的設備檔都是存放於 /dev 目錄或是其子目錄中。下列就是執行 System V 系統的設備檔案列表：

```
$ ls -l /dev
total 52
brw-rw-rw-  1 root     sys      51,  0 Aug 31 07:28 cd0          光碟機
brw-rw-rw-  2 bin      bin       2, 64 Feb 23  1997 fd0         預設的軟碟機
brw------   1 sysinfo sysinfo   1,  0 May  7  1996 hd00        第一部硬碟機
crw------   2 bin      bin       6,  0 Dec  5 14:12 lp0         印表機
cr--r--r--  1 root     root     50,  0 Aug 31 07:28 rcdt0        磁帶機
crw------   1 henry    terminal 0,  0 Oct 15 10:23 tty01        終端機
crw-rw-rw-  2 bin      bin       5,  0 May  7  1996 ttyla       串列埠 1
crw-rw-rw-  1 bin      bin       5,128 Feb 23  1997 ttylA      Modem 埠 1
```

SVR4 系統通常還有兩個目錄，包括/dev/dsk 與 /dev/rdsk，在這兩個目錄中，還會包含一些檔案。在上述各目錄中的檔案，有時在 /dev 目錄下，會有同等的設備檔（或是鏈結 link）。實際上您會在您的系統中，看到更多各式各樣的設備檔，舉例來說：您電腦的主記憶體之類。上面簡明的列表揭示了兩個重點：

- 設備檔可依權限欄位(permissions field)的第一個字元（b 或 c），區分為兩大類。
- 第五個欄位是由一對數字所組成；在其他檔案中，此欄通常顯示其檔案大小。而一個設備檔則是一對數字，設備檔不包含任何的資料。

  這些屬性的意義，會於稍後詳述。

21.1.1 區塊（Block）與字元（Character）設備

首先，讓我們瞭解磁碟的讀寫動作。當您下達指令儲存檔案時，存檔的動作會

與其他使用者的存檔請求合併，一起做硬碟整區塊的寫入。這裡的區塊所代表的是磁區(sector)的大集合。當您自硬碟讀取資料時，會先搜尋緩衝快取區(**buffer cache**)，看它最近是否有讀取過此資料。假如在緩衝快取區中有此筆資料，就能省下相當多的時間，不需再對硬碟作讀取動作。或者您不想透過此機制，直接就對此設備進行讀寫；而許多設備讓您能做這樣的抉擇，其讀寫的方式則能依呼叫的設備名稱相異而有不同。

如您先前注意到的，大多數於權限欄位的第一個字元不是 `c` 就是 `b`。軟碟機、光碟機與硬碟的權限欄位的第一個字元是 `b`。這些設備中的資料，都是以緩衝快取區做輔助，採用區塊的串列方式讀寫。所以此類設備被稱為區塊特殊設備(**block special devices**)。相對的，終端機、磁帶機與印表機就是字元特殊設備(**character special devices**)或稱原始資料設備(**raw devices**)，而在權限欄位的第一個字元就是 `c`，它們資料讀寫都不透過緩衝快取區，直接存取設備。


Note

在 System V 流派的作業系統中，許多設備都同時有 raw 與 block 兩種設備檔。如硬碟、軟碟與光碟機就是如此。大致上來說，區塊設備名稱之前加上 r，就是字元設備。區塊設備能於 /dev/dsk 目錄中找到，而相對的字元設備則是存於 /dev/rdsk 目錄中。

21.1.2　主要(Major)與次要(Minor)設備數字之意義

日常操作中，需透過設備驅動程式(**device driver**)方能使此設備運轉。當要對特定的設備動作時，系統核心會先呼叫正確的設備驅動程式，透過它將參數傳遞給此設備。所以系統核心不只需要知道設備的形式，對於此設備的詳細資料也要有所掌握，如軟碟機的磁片密度或硬碟切割的磁碟分割區數目等。

在之前的設備列表中，第五欄中之值並不是顯示檔案的大小；而是一對由逗號(,)分開的數字。這些數字分別被稱為主要(**major**)與次要設備數字(**minor device numbers**)。主要設備數字代表的是設備驅動程式；這實際上是此設備的形式。所有的硬碟如果都接在同一個控制器(controller)上，就都會有相同的主要設備數字。

次要設備數字，則代表核心程式要透過設備驅動程式傳遞的參數；通常也代表著此設備的特性。譬如，`fd0h1440` 與 `fd1h1440` 表示兩個軟碟機接在同一控制器上。所以兩者都有一樣的主要設備數字，但不同的次要設備數字。

設備檔一樣能有權限的控制。要將輸出傳送到終端機，您需要有對此設備寫入的權限；當您要讀取軟碟時，需要先擁有此設備檔的讀取權限。然而，大多數設備的權限與屬性，都只能由系統管理員設定。

21.2 設備名稱(Device Names)之意義

UNIX 系統設備檔具有同一設備能以不同檔名呼叫之特性。這是為了提供逆向相容(backward compatibility)或是能以特定的函數來結合各自獨立的設備元件。

表 21.1 列出的是 UNIX 與 Linux 系統中常見的設備檔名稱與其對應的實體設備。儘管在 Linux 各版本系統中都是使用一樣的設備名稱，但對 System V 系統而言，卻不是如此。 SVR4 流派的系統中，不同版本中的設備名稱也各異；假如您操作過 Solaris 或 HP-UX 系統，就會發現設備檔名稱是有差異的。

您於設備檔名稱中，會發現有許多 0、1 及其他數字。通常，您可以由設備檔名稱看出其對應的設備。System V 的設備檔是存放於 /dev/dsk 目錄下（原始資料設備存放於 /dev/rdsk 目錄）。如 /dev/dsk/f0q18dt 代表的是：(0)軟碟可開機、四倍密度，(18)代表是一個磁軌(track)區分為 18 個磁碟區段(sector)的1.44 MB 之 3.5 吋（區塊式，block）軟碟機。許多設備都有其預設的設備名稱；如上述的軟碟機就能以 /dev/fd0 直接呼叫。

表 21.1 （/dev 目錄中）常見的設備名稱

SVR4設備檔	Linux設備檔	代表之實體設備名稱
cd0 或 dsk/c0t6d0s2	cdrom	CD-ROM
fd0或diskette	fd0	預設的軟碟機
dsk/f0q18dt	fd0H1440	1.44MB 磁碟機
rdsk/f0q18dt	fd0H1440	1.44MB 軟碟機
hd00或dsk/c0t0d0s2	hda	第一部硬碟機
hd10或dsk/c1t3d0s2	hdb	第二部硬碟機
lp0	lp0	印表機
rcdt0或rmt/0	st0	磁帶機
term/1	tty1	終端機
tty1a	cua0	串列埠 1
tty2A	ttys1	數據機埠 2

在某些舊式的系統中，您可能看到 fd0135ds18 的設備檔，它所代表的是：(0)軟碟可開機、雙面、(135)代表有 135 條磁軌、(18)代表是一個磁軌區分為 18 個磁區的 3.5 吋磁碟機。您一樣能以原始資料設備方式呼叫，即/dev/rfd0135ds18。如在 XENIX 系統中就使用此設備檔名稱呼叫軟碟機，其他系統中為了提供逆向相容，也都有此設備檔。


Note

不同於一般的檔案與目錄，設備檔中不包含任何資料。它們都是直接指到實體的設備。

設備名稱與對應之設備

Linux 的設備檔是存放於 /dev 目錄中；設備檔名顯著不同：

```
lrwxrwxrwx   1 root  root       3 Mar 11 20:54 cdrom -> hdc
crw-rw----   1 root  uucp  5,  64 Nov 30 18:55 cua0          串列埠 1
brw-rw-rw-   1 root  root  2,  40 Nov 30 18:55 fd0H1440      第一部軟碟機
brw-rw-rw-   1 root  root  2,  41 May  1 04:19 fd1H1440      第二部軟碟機
brw------    1 root  root  3,   0 Nov 30 18:55 hda           第一部硬碟機
crw-rw----   1 root  lp    6,   0 Nov 30 18:55 lp0           印表機
lrwxrwxrwx   1 root  uucp      10 Apr 24 15:04 modem -> /dev/ttyS0
lrwxrwxrwx   1 root  root       9 Apr 24 15:04 mouse -> /dev/cua1
crw------    1 henry tty   4,   1 Apr  3 17:31 tty1          終端機
crw-rw----   1 root  uucp  4,  64 Nov 30 18:55 ttyS0          串列埠 1
```

Linux 系統中以 /dev/fd0H1440 代表 3.5 吋可開機的區塊式軟碟機；同時也藉由大量的連結，鍊結各設備檔。如您可發現 /dev/modem 是連結到 /dev/ttyS0（第一個輸出的串列埠）；滑鼠是連結到 /dev/cua1（第二個輸入的串列埠）；光碟機則是連結到代表硬碟機的(/dev/hdc)檔。

21.3　硬碟機

當您成為系統管理員，您需要製作磁碟分割區與檔案系統時，會需要瞭解硬碟機各組件的專有名稱。每部硬碟機中包含一至數片的雙面磁盤**(platters)**。每個磁頭則分別對碟片各面進行讀寫。也就是說，當有八面可供讀寫時，就需要八個磁頭。而磁頭是整組移動的，無法分別控制其移動。

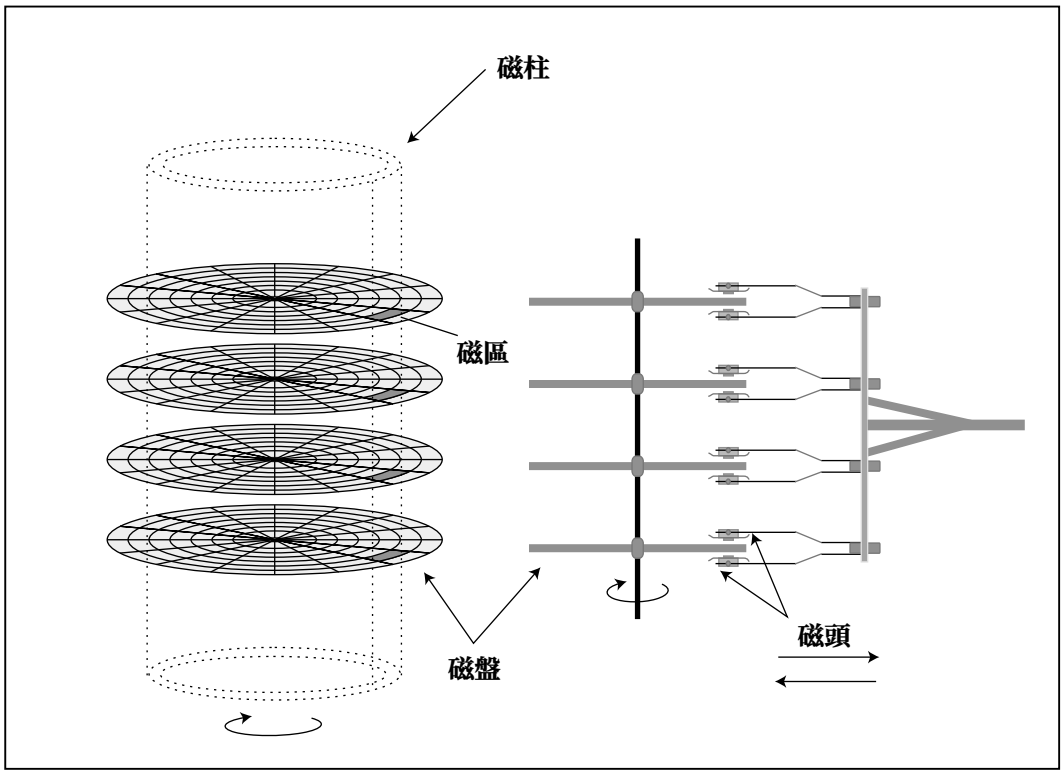
磁盤的每一面，都是由數條同心圓的磁軌**(tracks)**所組成。各面上的磁軌會帶有相同的磁軌編號；因此您可以想像在磁盤各面上，具有相同磁軌編號的磁軌形成磁柱**(cylinder)**。而各碟片有多少可用的磁軌，就會形成多少的磁柱。而磁軌又可在分割為數個磁區（**sector** 或 **block**）。因此如圖 21.1，當硬碟機中之磁盤有八面可供使用，每條磁軌可劃為 32 個磁區時，每個磁柱就共擁有 256 個磁區。

硬碟機是以固定的速度運轉（通常是每分鐘 3600 轉）。磁頭則是以放射狀似的動作於磁軌間移動；當磁頭移到欲讀取的磁軌時，所有的磁區於磁盤旋轉狀態下，在非常短時間內就能讓磁頭讀取完資料。

21.4　製作磁碟分割區(partitions)與檔案系統

如同其他的作業系統，UNIX 也需要先格式化硬碟機。格式化動作並不在作業系統管轄範圍。它會先將不良的磁軌標記，避免日後的讀寫動作使用到此問題磁

圖 21.1 硬碟機架構



軌。格式化通常是由供應商提供特殊的工具程式進行。今日，不論是 IDE 或 SCSI 的硬碟在出廠前，都已做好格式化；您就不用在花費精神在這方面。

要使用您的硬碟機前，需要先切割磁碟分割區**(partitions)**。每個分割區可被視為各自獨立的邏輯磁碟，以各自的設備檔呼叫。在您未於分割區製作一個或多個檔案系統前，您還是不能使用它。檔案系統不同於分割區，是一個包含各種 UNIX 系統組件的目錄結構。最後，您要確認的是所有的檔案系統要組成單一的UNIX 檔案系統，讓您能很容易的使用它。

21.4.1 磁碟分割區

分割磁碟(Partition)也是 BSD UNIX 與 Solaris 系統上磁碟切割**(slices)**的代名詞。從管理者的觀點來看，將硬碟分成數個切割區有下列優點：

- 獨立的分割區，能避免引起不同資料儲存區之間潛在的干擾、填塞。當使用者資料量突然間的暴增或持續的增加時，不至於影響到主要的系統分割區。
- 假如某一區域發生資料毀損，其他地區能藉此建立有效的屏障，隔離損壞區。讓系統管理員能針對受損區域進行修復，而不需關閉整部主機。

- 大量小塊的磁碟分割區意味著破碎區塊（fragmentation，請參閱 21.5.2 節）的產生，它應該被限制出現的量。過多的破碎區塊會影響到硬碟的整體效能。
- 假如系統事先進行適當的分割，每個磁碟分割區就能以一捲獨立的磁帶作備份。也就是說，管理者能針對各分割區，安排不同的備份時程。
- 多重開機系統（如 Linux 與 SCO UNIX 等）需要對不同的作業系統建有各自獨立之分割區。

當您在做硬碟分割時，您必須注意分割區的起始與結束，是否落在磁柱邊界範圍上。假如您從磁柱中間開始切割，硬碟的效能會因此降低。通常切割磁碟分割區之工具，會自動將它移到最近的完整磁柱群；如果沒有的話，就要勞煩您自己來處理。此外，當您用指定磁軌取代磁柱的劃分方式，假如您沒有事先規劃好，很容易就會超越磁柱之邊界範圍限制。


Note

多重開機系統（如 Linux 與 SCO UNIX 等），允許多種作業系統非同時地載入同一主機使用。在這種情形下，每個作業系統會需要一個獨立的磁碟分割區存放其系統。而系統會於開機時，提示使用者有多少作業系統可供起動。

21.4.2 檔案系統

在硬碟分割完成後，您必須在各分割區建立檔案系統**(file system)**。UNIX 系統通常是由許多各自起源於本身的根(root)並建有目錄樹之檔案系統組成。在您的主機上，沒有看到許多檔案系統的 “ root ”，是因為在使用時已經整合成一個大檔案系統。在瞭解如何建立這樣的檔案系統前，讓我們先來瞭解檔案系統的架構。

每個檔案系統都是由一連串 1024 個位元組的區塊所組成；通常分為四大組成元件：

- 開機區塊(boot block)——此區塊包含一個小的開機程式與磁碟分割區表。
- 超級區塊(superblock)——包含整個檔案系統全區域性(global)的資訊。此外，亦記錄了能目前能被使用的空白(free) inodes （請參閱 21.5.1 節）列表與資料區塊，讓系統核心在製造新檔案時，能立即拿來使用存放的區域。
- inode 區塊(inode block)——此區域包含整個檔案系統中每筆檔案的相關資訊表。除了檔案與目錄本身的名稱外，每筆資料的屬性都存放於此區。
- 資料區塊(data block)——作業系統檔、所有使用者建立的資料與程式，都存放於此。

檔案系統分布格式如圖 21.2 所示。在隨後的章節中，我們將仔細來看系統核心與上述各元件在儲存檔案時，如何一同動作找出存放空間。將這之前，我們先瞭解檔案系統的設備名稱命名格式的由來。

Note

當分割區中只有一個檔案系統時，*partition* 與 *file system* 這兩個字通常被視為同義詞。然而，也並不總是如此。某些系統如 SCO UNIX 就再將分割區切為 *divisions*；每個 *division* 各自擁有其檔案系統。Linux 系統也允許製作延伸分割區(extended partition)；各延伸分割區還有許多邏輯分割區(logical partitions)。如此，單個分割區就可能有多個檔案系統。

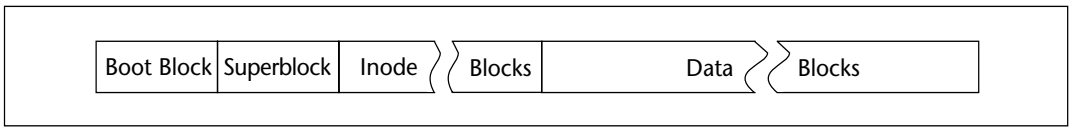
21.4.3 檔案系統設備名稱

一般而言，IDE 硬碟能支援切割到四個分割區，而 SCSI 硬碟則支援到八個分割區之多。每個檔案系統（與分割區）分別有其代表之設備檔，在不同的 UNIX平台版本中，檔名也都不太一樣。事實上，沒有任何地方像 UNIX 系統這般，在命名規範上有如此的分歧。

SVR4 流派的作業系統，使用 /dev/dsk 與 /dev/rdsk 目錄分別存放區塊與字元設備。它在命名結構上，包含有使用哪一個控制器編號(controller number)、控制器上的所佔用之設備位址資訊及其採用的分割區號碼。下列就是採用此規則的 Solaris 系統之命名範例：

- /dev/rdsk/c0t3d0s4——代表是第五個分割區(s4)、佔用第一個控制器(c0)的第一個設備位址(d0)。

圖 21.2 **檔案系統分佈格式**



- /dev/rdsk/c0t3d0s2——這裡是採用第三個分割區(s2)，但實際上就是整個硬碟。

就如上述，UNIX 系統中除了各分割區（或檔案系統）有其對應之設備檔外，還有一個特定的設備檔代表整個硬碟。在 Solaris 系統中使用的就是 2 號分割區（第三個分割區）來表示。但在其他的作業系統中，可能命名規則又有所不同。

Note

假如分割區中包含許多檔案系統，每一個檔案系統與分割區都會有各對應的設備檔。但依舊會有一個特定的設備檔代表整顆硬碟。

21.5 檔案系統元件

現在讓我們依組成檔案系統的四個主要元件之功能，用相反的順序進行解說。首先是 inode 與資料區塊，再來是超級區塊及開機區塊。由下而上的反向說明，能讓我們能較容易瞭解整個檔案系統的運作。

檔案系統之設備檔與對應之設備

Linux 系統之命名規則相當簡單；SCSI 硬碟以 sd 起始，而 IDE 硬碟則是以 hd 起始命名；隨後的就是磁碟機編號與分割區編號：

/dev/hda1——第一部(a)IDE 硬碟的第一個分割區
/dev/sdb3——第二部(b)SCSI 硬碟的第三個分割區
/dev/hdb——整個第二部 IDE 硬碟
/dev/sda——整個第一部 SCSI 硬碟

如上所見，Linux 系統也有一特定的設備檔對應到整部磁碟機。而在這檔名中就不包含任何的分割區編號。

如果 /dev/hda1 是一含有兩個邏輯分割區的延伸分割區，又該如何表示這兩個邏輯分割區呢？在這種情形下，這兩個邏輯分割區會分別對應 /dev/hda5、/dev/hda6 設備檔。在稍後解說 fdisk 工具時，會再進一步說明。

21.5.1 Inode 區塊

每個檔案都對應一個**inode**，即一個包含該檔案所有相關資訊（除了檔名）之 128 個位元組的列表。所有的 inodes 都是連續性地存放於檔案系統中使用者無法直接使用到的 **inode** 區塊。每個 inode 都包含下列的檔案屬性記錄：

- 檔案型態（一般檔案、目錄、設備等）。
- 連結到此檔案數目（此檔擁有的別名數目）。
- 檔案擁有者之 UID。
- 檔案擁有者之 GUID。
- 檔案權限設定（檔案之三組權限欄位）。
- 檔案所佔位元組大小。
- 檔案資料最後修改日期時間。
- 檔案資料最後讀取時間。
- inode最後修改日期時間。
- 指到檔案的十五個指標點之陣列。

inode 中之資料是藉由 **inode number** 之數字來讀取。在每個檔案系統中，每個檔案擁有一不重複之 inode 數字；它代表此 inode 資料於 inode 區塊存放之位置，而且經由簡單的運算就能找到真確位址。在 7.13 節曾介紹過，您能藉由 ls 加上 **-i** 參數來檢視 inode 數字。

您會發現不論是檔名與 inode 數字都不是存放於 inode 中；實際上，這兩筆資料是記錄於包含此檔的目錄檔中。而除了十五個指標點之陣列外，您都能藉由 ls 指令加上不同的參數來顯示 inode 記錄中的檔案屬性資料。至於檔案名稱，ls 指令則是從目錄檔取得。

當您開啟檔案時，它的 inode 資料會從硬碟被複製到維護在記憶體中系統本身的 inode 表中。系統核心在複製於記憶體的資料中（inode 緩衝區**inode buffers**）運作，而此區資料內容則是週期性的寫回硬碟。但因寫回硬碟的動作，並非與記憶體中的資料同步更新（記憶體中的資料會比較新）；所以當系統不正常斷電時，常會發生資料不一致的情形。當UNIX 系統本身無法回復時，就需要您指示檔案系統修復資料。


Note

inode **中記錄了除檔名以外的檔案各種屬性資料。檔名是記錄於存放此檔的目錄檔中。**UNIX **系統中將**inode **資料存放於檔案系統之目錄區域，它紀錄之** inode **架構遠比視窗系統的** FAT **檔案系統完整，**FAT**的資料儲存檔案各特性，包含修改時間、檔案大小及資料起始之磁柱位址等。**

21.5.2 資料區塊(Data Blocks)

大多數系統中，磁碟控制器所能讀寫的最小之區塊大小為 512 個位元組；通常也稱為實體區塊(**physical block**)。但系統核心是以一另稱為邏輯區塊(**logical block**)之區塊大小，進行檔案讀寫。許多 UNIX 工具軟體顯示其結果以實體區塊單位輸出，但實際上系統的效能是以您製作檔案系統時，訂定之邏輯區塊大小所決定。

不同於終端機、印表機這些字元設備一次讀寫一個字元；區塊設備如硬碟是以串列式或整區塊的資料一起作處理。標準的（邏輯）區塊大小從 1024 到 Solaris 系統中的 8192 個位元組都有。以整個區塊作讀寫運作來說，如果您以 1024 個位元組為最小區塊大小，要寫一筆 3 個位元組大小的資料時，就會有 1021 個位元組被浪費掉。而要維護一個組織良好與資料格式一致的檔案系統，就必須付出此代價。

資料區塊(**data blocks**)起始於 inode 區塊終止處。每一區塊都有其對應之位址，即一個數字位址代表資料於資料區塊中的真確位置。而包含資料的區塊亦稱為直接區塊(**direct blocks**)。

雖然各區塊是依序編號，但您很難找到一個經常遭到修改的檔案，會存放在連續區塊中。當一個檔案增大時，系統核心不一定能找到毗鄰有空的區塊可供使用；它會由硬碟中隨機取得四處散布的空區塊存放。這就造成了磁碟資料的破碎區塊，而導致讀寫效能降低。然而，此資料之不連續性有可能使檔案變得較大，或者變小。

因此，inode 就必須記錄所有直接區塊位址(direct block addresses)。然而您會發現這是不可行的，因為 inode 只能存放其部分資料（十二筆）。所以，檔案系統是以間接區塊來解決上述問題。這些區塊中只紀錄 inode 在第十三區段之前無法容納得下的直接區塊位址資料。而 inode 中就建立了一份間接區塊位址表，以取得實際的資料區塊位址；這就是我們稍後探索檔案所佔用之所有區塊的最佳指引。


Note

不論您系統中的邏輯區塊大小是 1024 **或** 8192 **個位元組，許多** UNIX **指令是用** 512 **個位元組的實體區塊**(physical block)**為單位輸出其結果。而在大多數系統上，**ls -s**、**df**、**du **與** find **指令只以實體區塊為磁碟空間計量單位。**

21.5.3 區塊位址架構(Block Addressing Scheme)

本節的討論是以 Solaris 系統使用的標準檔案系統(*ufs*)進行解說，但其概念亦能推及於其他檔案系統。下面將看到 inode 中的十五個磁碟區塊位址之陣列如何能記錄檔案所佔用的所有資料區塊。

前十二筆是記錄檔案的前十二個資料區塊位址。然而，假如檔案只有三個區塊大小，只有前三筆有記錄資料，其餘會都填上零。自第十三筆開始的三筆記錄，就不是指到直接區塊，而是間接區塊。如此錯綜複雜之架構，應用規則如下。

第十三筆記錄的是包含更多直接區塊位址之單一間接區塊(**single indirect block**)地址。當檔案大小再增加時，第十四筆指向的是雙重間接區塊(**double indirect block**)而它包含了單一間接區塊的位址。而最後的第十五個指標指向三重間接區塊(**triple indirect block**)，它指向雙重間接區塊。最後，所有的間接區塊就形成儲存檔案所有資料區塊的連結串列(**linked list**)。圖 21.3 對此資料區塊架構作一簡單之描繪。

在 32 位元機器上的 *ufs* 檔案系統中，最大的檔案大小是二十億個位元組(2G Bytes)。而今日 UNIX 系統在 Digital（現在的 Compaq）、Sun 與 IBM 的努力下已發展到 64 位元，使檔案大小限制已突破到數兆個位元組(Terabytes, 1 TB = 1000 GB)。

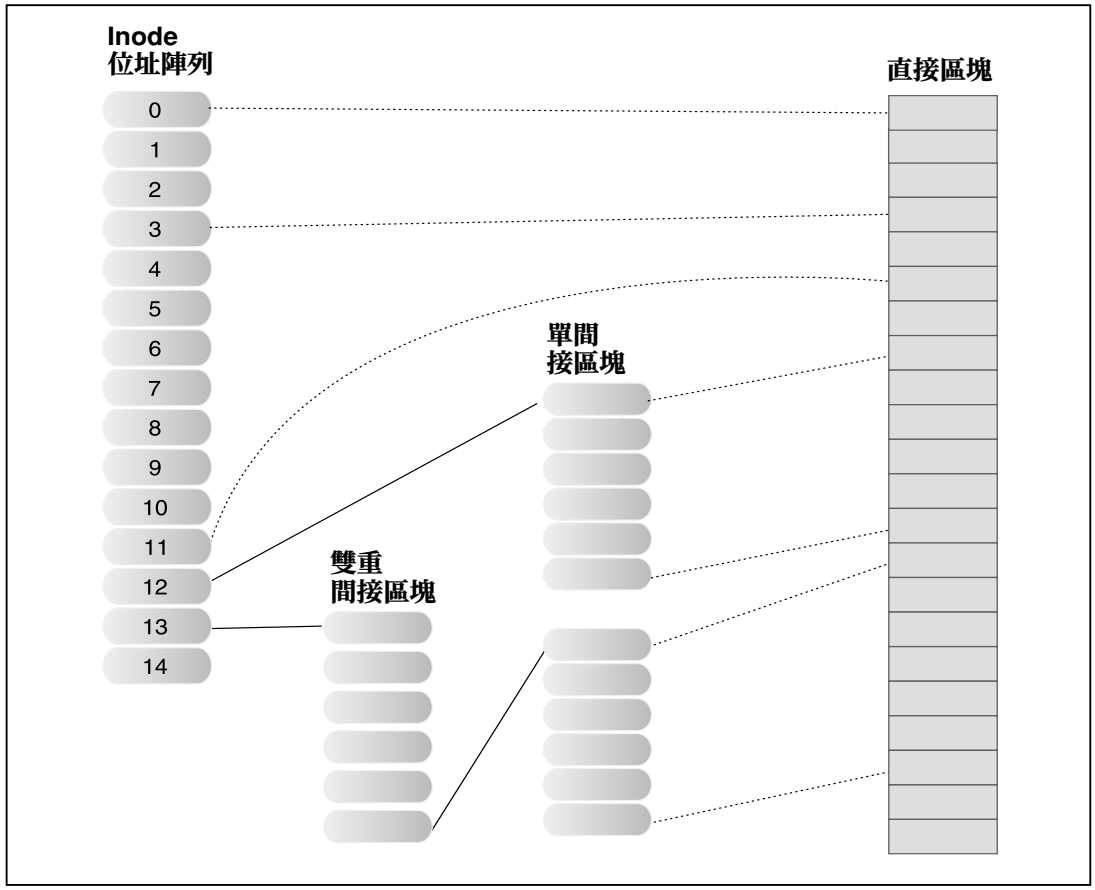
21.5.4 超級區塊（Superblock）

超級區塊係位於inode 區塊之前，即每個 UNIX 檔案系統之 「收支平衡表」(balance sheet)。它包含檔案全區域性的資訊包括硬碟使用情形、可利用的資料區塊與 inode。其資訊需由系統之正常操作隨時更新。下列是其主要內容：

- 檔案系統大小。
- 檔案系統之邏輯區塊長度。
- 最後修改時間。
- 可應用之空(free)資料區塊數與立即可用的空資料區塊列表。
- 可應用之 inode 數與立即可用的 inode 列表。
- 檔案系統狀態(”clean ” 或 “ dirty ”)。

就如同 inode，系統核心一樣將超級區塊資料複製到記憶體中。當在控制、更新 inode 與資料區塊位址時，就會對此複製資料進行讀寫。稍後系統核心會使用 sync 指令將複製於記憶體回存到硬碟中。因此雖然硬碟中不是最新的資料，但也

圖 21.3 **硬碟區塊位址架構**(Block Addressing Scheme)**——此圖中未展示三重間接**(Triple Indirect Block)**區塊部分**



相差不遠。而在系統關機前會再作一次此刷新動作，以確保資料確實更新完畢。


Note

UNIX 系統在超級區塊受損時，會無法開機。許多系統（如 Solaris **與** Linux**）在硬碟各不同區域中，複製有超級區塊資料，以解決此問題。當超級區塊資料損壞時，系統即可被指示轉由其他備份取得資料，繼續運轉。**

21.5.5 開機區塊(Boot Block)

繼續進行我們的檔案系統反向解說，終於來到了最起始之區塊。位於超級區塊前的就是開機區塊。Linux 的使用者可能會回想起在安裝系統時，會看到 LILO 選項提到主要開機記錄區(**Master Boot Record，MBR**)，指的就是這裡。開機區塊中

除了磁碟分割區資料表(partition table)外，尚含有一個小的開機(bootstrapping)程式。

當系統開機時，系統的 BIOS 先檢查第一顆(Master) 硬碟是否存在，之後就將其開機區塊整段資料載入記憶體中。然後就將控制權交由開機區塊內之開機程式處理。開機程式隨後會將系統核心程式（**unix**、**genunix** 或 **vmlinuz** 檔）載入記憶體。開機程式是由主(root)檔案系統內的開機區塊提供，因而其餘檔案系統的開機區塊都是沒有資料保持空白。

21.6　目錄檔(Directory)

目錄檔中包含兩個檔案資訊，即檔名與 inode 數字。因 inode 本身無法經由此 inode 數字取得，因此可以說是目錄與 inode 之間形成一關連式資料庫，藉由 inode 數字建立此兩列表(tables)之間的關係。以關連式資料庫(RDBMS)術語來說，您能藉由連結(joining)兩列表之個別獨立之列(rows)，得到檔案屬性資料。

當您新增一個檔案的鏈結(link)，它的 inode 之鏈結數記錄會隨著遞增。目錄檔中也會增加一新檔名，而 inode 數字資料會由原始檔複製過來。而當您以 `rm` 指令刪除鏈結檔時，inode 之鏈結數記錄也隨著減少，而目錄檔中會移除此鏈結檔的資料。所以當inode 之鏈結數記錄歸零時，代表此檔案被完全刪除，其存放資料的資料區塊稍後就會釋放出來，新增到空白可使用區表列中。

21.6.1　系統核心程式如何讀寫檔案？

根據上述解說，您已能瞭解系統核心程式在存取檔案和目錄時所使用的地址架構。當您執行 `cat foo` 指令時，會發生什麼狀況？檔案會被找出並顯示於標準輸出介面，之後系統核心必須確認檔案的結束位址，以停止繼續顯示。完整之運作過程如下：

1.通常系統核心會在記憶體中，建立一使用中目錄之 inode 數字。藉此搜尋 inode 區塊並找出目錄之 inode 位址。

2.由 inode 取得包含目錄檔的資料區塊位址。

3.系統核心藉由目錄檔，讀取 foo 檔之 inode 數字。

4.之後再回到 inode 區塊取得 **foo** 檔之 inode 資料。

5.系統核心讀取檔案大小與硬碟位址，並到所有的間接區塊（如果有），讀入所有相關的直接資料區塊位址資訊。

6.最後，驅動硬碟機磁頭到儲存資料之區塊，讀取先前取得的檔案大小位元組之後結束。並沒有其他的方式，能知道是否已讀完所有檔案資料。


Tip

您應該盡量減少於一目錄中，存放太多檔案而佔據大量（甚至可能是不連續）的區塊。這會降低系統核心運作效能與讀寫檔案效率。因此您應該再多建立幾個目錄，將檔案盡可能分開存放。

21.7 標準的檔案系統(Standard File Systems)

先前討論都是單一的檔案系統。但實際上，大多數的 UNIX 系統會將硬碟分成八個分割區。此外，假如您有許多硬碟，每個硬碟上至少都會建立一檔案系統。通常來說，UNIX 系統中都有這兩個檔案系統：

- *root* 每個 UNIX 系統都擁有此檔案系統。它包含 UNIX 系統最基本之組件，即 根目錄、/bin、/usr/bin/、/etc、/sbin、/usr/sbin、/dev 與 /lib 目錄，及使系統能正常運轉的各種工具軟體。當系統以 single-user 模式開機時，系統管理者能應用的唯一檔案系統就是 root 分割區。
- *swap* 每個系統中都應有一 swap 檔案系統，供系統核心暫時處理各異動的程序之資料。當系統記憶體使用頻繁時，系統核心會將部分程序由記憶體暫時移至此檔案系統。之後當被置換的程序要繼續執行時，會再載回記憶體中。而使用者無法直接對此檔案系統進行讀寫。

UNIX 系統（特別是較龐大的系統）會建置許多檔案系統。系統的檔案應該要與使用者檔案分開放置，它們各有其獨立的檔案系統以供存放。Linux 與 SVR4 系統將使用者的家目錄置放於 /home 目錄下，部分較舊的系統則是存放 /usr 目錄。因此 /home 目錄通常會另外建立獨立之檔案系統。假如 /home 沒有多餘的空間可供使用，系統管理者通常會另外建立 /usr2 或 /u 目錄。

您也可以建立 /tmp 與 /var/tmp 這兩個額外的檔案系統，這樣暫存檔與系統記錄檔就不會無限制成長，影響到其他重要資料存放。假如該主機需要處理大量的電子郵件，建議您為 /var/mail 目錄建置獨立的檔案系統。當管理者需要安裝資料庫軟體，也應該建立另外的檔案系統，如建立 /oracle 檔案系統以安裝 Oracle（甲骨文） 關連式資料庫。

21.8 檔案系統類別(File System Types)

一開始，UNIX 世界只有 AT&T 與 Berkeley 建置的兩種檔案系統。但隨著 UNIX 系統的發展，越來越多的檔案系統被創造出來。因此，您需要瞭解下列幾種您可能遭遇到的檔案系統：

- *s5* 在 SVR4 出現前，這是 System V 使用的唯一檔案系統；但今日 SVR4 僅藉此名稱提供逆向支援。此檔案系統只有一個超級區塊，使用的邏輯區塊大小為 512 或 1024 個位元組。而且無法處理檔名長度超過十四個字元的檔案。

- *ufs*　因檔案系統的區塊大小可達 64K 個位元組，一般認為其效能較 *s5* 佳。所以此 Berkeley 快速檔案系統(Berkeley Fast File System)得到 SVR4 系統採用與大多數的 UNIX 系統支援。而它在每個磁柱群組中都存放一筆超級區塊資料；並且不同於 *s5*，它支援達 255 字元的長檔名、符號鏈結與硬碟空間配額(disk quotas)管理。
- *ext2*　這是 Linux 採用的標準檔案系統。它使用的區塊大小為 1024 個位元組；另外如同 *ufs*，使用多重超級區塊備份並支援符號鏈結。
- *iso9660* 或 *hsfs*　此為 CD-ROM 使用的標準檔案系統。它採用 DOS 模式的 8+3 檔名格式。自從UNIX 系統支援長檔名後，*hsfs* 也提供 Rock Ridge 擴充（ISO9660 之延伸，支援大於八個字元長檔名與其他功能）來容納長檔名。
- *msdos* 或 *pcfs*　大多數的 UNIX 系統都支援 DOS 檔案系統。您可以在軟碟機建立此格式之檔案系統，然後藉此將檔案傳遞給視窗作業系統（微軟）。Linux 與 Solaris 系統能直接讀寫硬碟中的 DOS檔案系統資料。
- *swap*　前面過章節已介紹。
- *bfs*　開機之檔案系統。SVR4 流派系統以此存放開機程式與 UNIX 系統核心。使用者不能直接讀取此檔案系統。
- *proc* 或 *procfs*　這可被視為記憶體所維護的虛擬檔案系統(pseudo-file system)，檔名就是各 PID（程式識別碼），而看起來好像檔名的物件，事實上卻不是真正的檔案。它存放每個執行中的程式自記憶體中映射下來之資料，以供系統讀取每個程式資訊。而使用者也能在此取得大多數的程式資訊包括它們的 PID。

無疑的，上述的某些檔案系統尚提供其他特殊功能，這部分我們暫不討論。除了上面介紹的檔案系統，各 UNIX 的軟體製造商都提供他們自己的檔案系統，但他們通常都會支援大多數上面所討論過的檔案系統。您能藉由指令（如 `mkfs` 及 `mount`）加上描述此檔案系統的選項參數來使用該檔案系統，這就是為什麼您要瞭解這些您常用的檔案系統格式。

21.9　製作分割區(Partitions)與檔案系統

UNIX 系統的多樣性，在您建立分割區時就能感受到。不同版本的 UNIX 會應用不同的工具切割硬碟。雖然大致上來講原理都一樣；但有時某些細部操作步驟不同，導致有相當的差異。系統管理員應該不會允許您在她的系統上作增、刪切割區的實驗，因此下面我們就以 Linux 系統來講解。

21.9.1　使用 `fdisk` 指令製作分割區

Linux 與 SCO UNIX 系統在 Intel 主機(PC)上，都允許使用者建立多重作業系

統。無庸置疑的，它們都提供有如視窗系統模式的 fdisk 指令以建立、刪除與啟動分割區(activate partitions)。然而在 Linux 系統的 fdisk 指令之操作與視窗系統並不相同。

首先於第一部 IDE 硬碟中，我們將建立一分割區以安裝 Oracle 資料庫。執行 fdisk 指令後，使用 p 內部指令就能看到如下的分割區表：

```
Disk /dev/hda: 255 heads, 63 sectors, 784 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1       217   1743021    5  Extended
/dev/hda2           218       478   2096482+   6  DOS16-bit >=32M
/dev/hda4           620       784   1325362+  83  Linux native
/dev/hda5             1         5     40099+  82  Linux swap
/dev/hda6             6       217   1702858   83  Linux native
```

由上列資訊得知，此硬碟有 16065 × 512 位元組大小的磁柱群784 組。因此這是顆 6.5 G 個位元組大小(784 × 16065 × 512)且包含許多分割區的硬碟。Linux 分割區與 DOS 下的分割區相當類似。就如前所描述，您能建立一主分割區(primary partition)並在其中建立一檔案系統；或是建成能包含許多邏輯分割區(logical partitions)的延伸分割區(extended partition)。

您可能注意到第一個分割區(**hda1**）與第五(**hda5**)、六(**hda6**)個分割區都用到同一磁柱區。(**hda1**)實際上就是包含兩個邏輯分割區的延伸分割區。其中(hda5)分割區是系統的 swap 分割區，而另一個(**hda6**)則是安裝 Linux 系統。每一個分割區大小都是以 1024 個位元組大小的區塊(Blocks)表示。而 active 分割區則於第二欄位以*****標示。

您可以看到硬碟上還有安裝有其他作業系統。第四個分割區(**hda4**)是另一個 Linux 系統，而第二個分割區(**hda2**)則是執行視窗作業系統的 DOS 檔案系統。我們可將 **hda5** 與 **hda6** 視為 **hda1** 的 「子集(subset)」，所以於此硬碟就有三個 「頂層(top-level)」之分割區。自 479 到 619 有未使用的磁柱群組，我們能利用此空間以建立第三個分割區。

fdisk 之內部指令m 可以列出所有可供我們作業時所需的各內部指令功能集：

簡略列示

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   w   write table to disk and exit
```

要建立分割區，您能藉 n 內部指令，提供分割區編號及起始與結束的磁柱編號：

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)                    請注意此項
p
Partition number (1-4): 3
First cylinder (479-784, default 479): 479
Last cylinder or +size or +sizeM or +sizeK (479-619, default 619): 619
```

Linux 會自動建議您預設的磁柱值；但請您自己運算後，再填入正確的磁柱編號。現在，再使用 p 內部指令檢視新增的磁碟分割區：

```
   Device Boot  Start     End   Blocks   Id  System
/dev/hda1   *       1     217  1743021    5  Extended
/dev/hda2         218     478  2096482+   6  DOS 16-bit >=32M
/dev/hda3         479     619  1132582+  83  Linux native          新增之分割區
/dev/hda4         620     784  1325362+  83  Linux native
/dev/hda5           1       5    40099+  82  Linux swap
/dev/hda6           6     217  1702858   83  Linux native
```

現在我們已建立一大小超過 1GB 的新磁碟分割區。在修改過分割區表後，我們須以內部指令 w 將資料寫回開機區塊。之後程式會提示您重新啟動系統，讓新設定的檔案系統能夠運作。


Tip

要在同一個 Linux 分割區建立多個檔案系統；您首先需建立一延伸分割區，然後在此分割區製作多個邏輯分割區。假如您覺得以 fdisk 指令難以完成上述動作，您能嘗試另一選單模式的 cfdisk 程式。

21.9.2 使用 mkfs 指令製作檔案系統

在建立磁碟分割區後，您需要在此分割區上製作檔案系統後，才能使用該分割區。儘管今日有許多前端工具可供使用，最常被使用的還是 mkfs 指令，即最普遍的檔案系統製作工具。基於已在上面的 Linux 系統建置的分割區範例中，我們使用 mkfs 指令並加上 **-t** 的參數，以指定欲製作的檔案系統類型：

```
# mkfs -t ext2 /dev/hda3
mke2fs 1.14, 9-Jan-1999 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
283560 inodes, 1132582 blocks
56629 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
139 block groups
```

```
8192 blocks per group, 8192 fragments per group
2040 inodes per group
Superblock backups stored on blocks:
      8193, 16385, 24577, 32769, 40961, 49153, 57345, 65537, 73729, 81921,
```

上述指令會製作一 *ext2*類型的檔案系統，即使用 1024 個位元組區塊大小的標準 Linux 檔案系統。而每一個複製的超級區塊是間隔 8192 個區塊來配置。*ufs* 與 *ext2* 檔案系統都是採用 *fragments* 技術以節省空間的浪費，探討此部分已超越本書的範圍，假如您有興趣請參閱由O'Reilly & Associates 出版Mike Loukides 所著之 *System Performance Tuning* 一書。

在 Linux 系統中的 mkfs 指令就好像是mke2fs的前端工具一般，mke2fs 是實際執行製作檔案系統的工作。因為我們沒有設定區塊值，mkfs 會以其預定值建立 inode 與資料區塊。因此兩區塊使用的空間是共用的，所以當其中之一填滿時，整個檔案系統就被裝滿，無法再填入新資料。不過一般而言，除非您的檔案系統中需處理極大量的小型檔案，您不必更動這些預設值。


Tip

您最好將 mkfs 指令輸出之複製的超級區塊存放之區塊位址給記錄下來。當主要的超級區塊受損時，您就能藉 fsck 指令指到另一替代的超級區塊以修復硬碟。


Note

Solaris 系統是採用選單模式的 format 指令來切割分割區。在 format> 提示符號下，您需要以 partition 指令進入分割區選單。在這個選單中，您能修改、設定最多到八個分割區，而分割區 1 與 2 （第二個與第三個）分別保留給 swap 的分割區與代表整個硬碟的分割區名稱。之後您需要以 label 指令將更動的分割區表的資料寫回硬碟。而 Solaris 與 HP-UX 都是使用 newfs 指令當作是 mkfs 的前端命令：

```
newfs /dev/rdsk/c0t3d0s0
```

21.10 掛上(Mounting)與卸下(Unmounting)檔案系統

在前面幾節執行步驟下，我們已擁有四個標準元件及包含一空的根目錄與一個 inode 區域的檔案系統。然而，root 檔案系統（安裝系統時所建置）並不知道此新增的檔案系統已存在。而這新的檔案系統要在您將它掛上 root 檔案系統（可能是 hda4 或 hda6）之後，才能開始存放資料。

掛上檔案系統的動作稱為 mounting，即檔案系統將它自己掛上 root 檔案系統的不同掛入點的過程。連結動作發生的位置點稱為掛入點**(mount point)**。在您掛上其他的檔案系統後，root 檔案系統就成為主檔案系統，它的 root目錄就成為整合過的檔案系統之根目錄。

21.10.1　使用 mount 指令掛上檔案系統

您需藉由 mount 指令來掛上檔案系統。在您掛上一個新檔案系統並執行此指令時，此指令需要兩個參數，包括檔案系統的設備名稱與檔案系統要被掛入的目錄名稱。而在您掛上檔案系統前，需要先在主檔案系統下建立一空目錄（例如：**/oracle**）。新檔案系統的根目錄就是要被掛在此目錄之下。

mount 指令使用一選項以提供檔案系統格式。此選項在不同的UNIX 版本可能有所不同，以下是我們如何以 mount 指令將檔案系統掛上 Solaris 與 Linux 系統：

```
mount -F ufs /dev/dsk/c0t3d0s5 /oracle                    Solaris
mount -t ext2 /dev/hda3 /oracle                             Linux
```

在掛上主檔案系統後，原先由mkfs 指令建立的根目錄就失去其實質意義。當它附屬於主檔案系統時，就是以 **/oracle** 目錄的身份出現。這讓使用者在使用此整合的檔案系統時，會認為只使用一個檔案系統一樣；在使用者於 **/oracle** 與 **/home** 目錄作資料轉移時，根本感覺不到是兩部硬碟機在運作！

上面兩個掛載檔案系統的範例，其新增的檔案系統都是預設類型，因此並不需要再加上 **-F** （或 **-t**）選項。假如您要掛上其他的檔案系統類型，就需要加上該選項：

```
mount -F hsfs -r /dev/dsk/c0t6d0s0 /cdrom          CDROM—Solaris
mount -F pcfs /dev/diskette /floppy                DOS 軟碟機—Solaris
mount -t iso9660 /dev/cdrom /mnt/cdrom             CDROM—Linux
mount -t vfat /dev/hda1 /msdos                     視窗系統硬碟—Linux
mount -t msdos /dev/fd0 /floppy                    DOS 軟碟機—Linux
```

mount -a 會將嘗試將設定檔中列出的所有檔案系統掛上。單獨使用 mount 指令本身，會列出目前已掛上的檔案系統：

```
# mount
/ on /dev/dsk/c0t0d0s0 read/write/setuid/largefiles on Thu Apr 20 10:00:10 2000
/proc on /proc read/write/setuid on Thu Apr 20 10:00:10 2000
/dev/fd on fd read/write/setuid on Thu Apr 20 10:00:10 2000
/oracle on /dev/dsk/c0t0d0s3 setuid/read/write/largefiles on Thu Apr 20 10:00:15 2000
/u01 on /dev/dsk/c1t3d0s1 setuid/read/write/largefiles on Thu Apr 20 10:00:15 2000
/u02 on /dev/dsk/c1t3d0s3 setuid/read/write/largefiles on Thu Apr 20 10:00:15 2000
/u03 on /dev/dsk/c1t3d0s4 setuid/read/write/largefiles on Thu Apr 20 10:00:15 2000
```

mount 指令是從 **/etc/mnttab**(Solaris)或 **/etc/mtab** (Linux) 檔中取得上列資訊。接下來我們將對 mount 設定檔作簡短的介紹。


Note

通常是用空目錄來當掛入點，但如果該目錄下原先就有檔案，這些檔案在掛載動作完成時，會暫時消失。而在卸下檔案系統時，檔案才會再出現。

Linux 系統對先前使用 8+3 格式檔名的 DOS 檔案系統(*msdos*)與支援長檔名的視窗作業系統 95/98/2000 等檔案系統(*vfat*)，各以不同的檔案系統類型支援。因此在您掛上一 DOS 磁碟後，發現當中的檔名有 （波浪符號，tilde）符號出現，那您一定是將支援長檔名的 *vfat* 檔案系統，錯以 *msdos* 格式掛上主檔案系統。

21.10.2 使用 umount 指令卸下檔案系統

在卸下檔案系統時，要由 umount 指令（請注意其拼字）來達成，它需要檔案系統設備名稱或掛入點名稱當參數，以卸載此檔案系統。以先前建置與掛上的檔案系統為例，我們能以下前兩種方式之一卸下此檔案系統：

```
umount /oracle                          指定掛入點名稱
umount /dev/hda3                        或設備名稱—Linux
umount /dev/dsk/c0t3d0s5                Solaris
```

當您已開啟檔案系統中的任一檔案時，是無法卸下此檔案系統的。進一步來說，除非您到某目錄的上層，否則您無法將此目錄移除一樣；在您依舊位於此檔案系統時，無法將該檔案系統卸下。假如您嘗試以上述的情形作測試，您將會看到下列的訊息：

```
# umount /dev/hda3
umount: /oracle: device is busy
```

您也能以 umount 指令同時卸下所有的檔案系統或依檔案類型來卸下：

```
umount -t ext2                          只卸下所有的 ext2 檔案系統—Linux
umount -a                               卸下所有檔案系統
```

選項 **-a** 會卸下目前已掛上的所有檔案系統，除了執行系統必須的檔案系統之外。已掛上的檔案系統，其資料是從 **/etc/mnttab**(Solaris)或 **/etc/mtab**(Linux)中取得。

21.10.3 mount 選項 (-o) 與 /etc/fstab 檔簡介

mount 指令能於 **-o** 選項後，加上許多依檔案系統特性不同所提供之特定功能設定；如果要加上的功能設定參數不只一個，您能以逗號(,)作分隔。下列範例中，雖然都是用 Linux 設備名稱，但大多數的參數可以通用在大部分系統上：

```
mount -o ro /dev/sdb3 /usr/local                唯讀
mount -o exec /dev/cdrom /mnt/cdrom             允許執行二進位檔
mount -o rw,remount /dev/hda3 /home             以讀寫模式重新掛上檔案系統
```

上面第一個例子是將檔案系統以唯讀模式掛上；此外 mount 指令以 **-r** 選項也能做到一樣的效果。第二個例子中，讓您能在 CD-ROM 中直接run執行檔(**exec**)。最後的範例是，當您要將原先以唯讀模式掛上的檔案系統改成讀寫模式時，您不須將檔案系統卸下再掛上，您只要合併使用 **rw** 與 **remount** 參數就能達到目的。

mount 指令使用一個設定檔；當您執行mount -a 指令時，所有在設定檔中的檔案系統就會依序掛上。系統啟動時就是執行上述動作，因此在登入時，您會發現系統已掛上可用的檔案系統。通常此設定檔為 **/etc/fstab**，但在使用 SVR4 流派的 Solaris 系統是使用不同格式的 **/etc/vfstab** 設定檔。它們在參數設定上也有一些小小的差異。下面我們介紹的是 Linux 系統使用之 **/etc/fstab** 檔中的幾行內容：

```
# Mount device  Mount Point   File System Type    mount Options
/dev/hda5       swap          swap                defaults       0       0
/dev/hda6       /             ext2                defaults       1       1
/dev/hda2       /dosc         vfat                defaults       0       0
/dev/hda3       /oracle       ext2                defaults       1       2
/dev/hdc        /mnt/cdrom    iso9660             ro,noauto,user 0       0
/dev/fd0        /floppy       auto                noauto,user    0       0
none            /proc         proc                defaults       0       0
```

設定檔中每一行對要掛上的檔案系統，都加上了需要的相關參數。表列中除了各硬碟外，還掛上了軟碟機與光碟機。前面四欄您能很容易瞭解其意義，其中 **noauto** 選項是指該檔案系統即使在執行 mount -a 指令時，都不會自動掛上，而是需要由您手動來完成此動作。

而當您在 **/etc/fstab** 檔中已指定 CD-ROM（或其他設備）要掛上時所需參數，您再手動掛上時，就只要在mount 指令之後填上設備名稱或掛入點其中之一即可：

```
mount /dev/hdc
mount /mnt/cdrom
```

mount *指令會自* /etc/fstab *表中取得所需參數，自行補上*

user 選項讓您能以一般使用者的身份來掛上此設備；而軟碟機的 **auto** 檔案類型，讓其磁碟片於掛上系統時，mount 會由 **/proc/filesystems** 中所列的資訊，自動嘗試各種可能的檔案類型。

而第五個欄位上的 **1**，代表的是此檔案系統須由 dump 指令進行備份。第六個欄位則是當系統啟動時，檔案系統要被 fsck 指令檢查的順序。


Note

mount -a 指令如同系統啟動時一般，會將設定檔 /etc/fstab 或 Solaris **系統的 /etc/vfstab 中的檔案系統一併掛上。相同的** umount -a **指令會將** Linux **之 /etc/mtab 或** Solaris**的 /etc/mnntab 檔中紀錄的檔案系統全部卸下。而某些系統如** Solaris **系統還有提供** mountall **與** umountall **指令（實際上是** Shell Scripts**），讓您**

能簡單地掛上或卸下所有檔案系統。

21.11 以 fsck 指令檢視、修復檔案系統

延緩將記憶體中的複製資料更新到磁碟之超級區塊與 inode 區塊的 UNIX 內建功能（請參閱 21.5.1 節），雖然對效能提升有一定的幫助，但也導致資料不一致的現象出現。假如在硬碟寫回超級區塊資料前就被斷電，檔案系統就會喪失其完整性。有許多種資料不一致的情形，會導致整個檔案系統資料毀損；下列就是最常見幾種：

- 兩個以上的 inode 指到同一磁碟區塊。
- 某區塊被標示為可使用(free)區塊，但超級區塊中卻沒有此筆可利用空間的資料。
- 使用中的區塊被標示為無資料的可使用區塊。
- 一個 inode既不是被標記使用中也不是可使用的，或是擁有超出正常範圍的錯誤區塊編號。
- inode 記錄的檔案大小與資料位址陣列記載的資料區塊數不符。
- 受損的超級區塊中有太多謬誤的統計資料。
- 檔案不屬於任何目錄，或於 inode 中記錄的檔案類型錯誤。

fsck （檔案系統一致性檢查，file system consistency check）指令，就是用來檢查與修復受損的檔案系統。此指令通常扮演著一些真正執行檔案系統特定工作的工具程式（如 fsck_ufs 或 fsck.ext2）之前端(front-end)，它被用來修復無法掛上的檔案系統。

包含 Solaris 之許多系統，檔案系統上都有做 “ dirty ” 或 “ clean ” 的標示。fsck 指令通常只對被標示為 “ dirty ” 的檔案系統並於下次系統啟動時作檢查。您也能在前面提及之 mount 設定檔 /etc/fstab（Solaris 是 /etc/vfstab）中的第六個欄位指定檔案系統被檢查時(fsck)的順序，即由主檔案系統的root開始。而 fsck 指令也能依下列方式，針對某檔案系統作檢查：

```
# fsck /dev/rdsk/c0t3d0s5
** /dev/rdsk/c0t3d0s5
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
```

fsck 指令會分成上列五步驟依序進行，上面的輸出表示檔案系統是一致的。當檔案系統損毀時，會有檢查結果訊息與修復問題提示出現在系統的主控台（console）。而您必須要在修復問題提示上提供正確的回答。下列解說 fsck 指令在每一

個階段上的工作：

- *Phase 1*　檢查 inode 格式是否正確及是否記錄不正確的區塊數或重複的區塊。fsck 會將其發現記載超越正常範圍區塊數之區塊標示為 **BAD**，被別的inode 重複指向的區塊標示為 **DUP**。
- *Phase 2*　檢查自根目錄起的所有目錄檔，及 *Phase 1* 中偵測出超越正常範圍(OUT OF RANGE)的 inode 數。fsck 會嘗試將有問題之目錄或檔案全部移除，以修正錯誤。
- *Phase 3*　檢視未被參考到的目錄，將其所有檔案存到 **/lost+found** 目錄中，以待稍後之測試。存放於此的檔案都是以其 inode 數字為檔名，而您必須確定每一個檔案系統中都先建有 **/lost+found** 目錄，因為 fsck 在執行時，並不會自行建立此目錄。
- *Phase 4*　檢查目錄檔中存放的 inode 之鏈結數，然後依受損程度提示您要移除或由 **/lost+found** 目錄中的資料重新連結。稍後 fsck 會將它計算出的可用(free) inode 總數與超級區塊中的記錄作比較。
- *Phase 5*　最後，在上面階段fsck 所找到的空區塊總計與超級區塊記錄的數字作比較，它會提示您是否要將最新檢查出的空區塊資料更新到超級區塊，以完成整個修復動作。

當您不加其他選項執行 fsck 時，在修復它所發現的任何損壞時都徵求您的確認。不過大致上來說，對每項確認您都可以回答 **y**，很安全的讓 fsck 自行修復。（譯註：在檔案系統毀損時，**fsck** 是您拯救資料的唯一工具，即使會損失部分資料，比起損失整個檔案系統的資料，也是值得的。）您也能藉 fsck -y（Linux 中的 e2fsck 是加上 **-p**）讓系統自動回應 **y**，減少檢查的等待時間。另外在 Solaris 系統上提供之 fsck -m 可以只對檔案系統作檢查，不做任何修復動作。

fsck 指令能修復、檢查區塊與字元設備，但檢查字元設備會稍快些。因此系統管理員最好先卸下檔案系統再做檢查，以增進其效率。但因主檔案系統(root file system)於正常操作時無法卸下，因此建議您於單人操作模式(single-user mode)下檢查。

但如果超級區塊已損毀至無法修復的地步，您就需在執行 fsck 加上 **-b** 選項（Solaris 系統中則是用 **-o b=*n***，***n*** 代表備份的區塊位址）以指定另一個超級區塊位址。而此位址就是來自您執行 mk**fs** 製作新檔案系統時（Solaris 系統中執行 newfs -Nv 也能產出一樣的列表），所抄寫下之位址列表。但有時整個檔案系統會損壞到無可救藥的地步，簡單的 fsck 修復動作已經無法挽回受損資料時，重新安裝整個系統就成為您唯一的選擇。


Tip

fsck 經常會將它懷疑不完整的檔案改以 inode 數字為檔名後，移到 /lost+found 目錄中。您應該寫下這些 inode 數字，以當作日後磁碟修復時的參考。此外如果在系統當機後，發現有檔案遺失，您可以先到 /lost+found 目錄中找找，有可能是被 fsck 移到此處。

21.11.1 sync 之問題

系統的update服務程式會呼叫 sync 程式每三十秒將暫存(cache)於記憶體中的超級區塊、inode 與資料區塊資料，寫回硬碟中。當您手動執行此指令後返回提示符號時，代表刷新資料的動作已排入系統執行行程中，而且第二次的 sync 動作在第一個完全執行完之後，才會繼續動作。

當 fsck 重建檔案系統後，複製在記憶體中的超級區塊與其他列表有可能包含舊的或錯誤的資料。在這種特殊情形下，硬碟中的資訊可能比記憶體中複製的部分還要新。此時 fsck 就會出現下列訊息：

```
***** BOOT UNIX (NO SYNC!) *****
```

通常在 root 檔案系統發生嚴重問題時，您就有機會看到這樣提示訊息。它提示您假使您又執行 sync 或 /sbin/shutdown 指令將記憶體中錯誤的超級區塊資料寫回硬碟，那麼先前 fsck 修復的部分就會被覆蓋消失。相反的，您應該盡快壓下 reset 鈕直接重新啟動系統（Solaris 系統中，您能以 **reboot-n** 指令不做 **sync**動作，直接重新開機）。而這動作需要在系統自動驅動 sync 刷新資料前完成，才不會將錯誤資料再回存硬碟中。

摘　要

所有設備檔都儲存於 **/dev** 目錄中。設備檔的 inode 都包含一對數字（主要 major 與次要 minor）。主要設備數字代表設備驅動程式，即裝置的類型。次要設備數字則是系統核心程式要傳遞給設備驅動程式的一連串參數。

設備依資料讀取與寫回格式，可區分為區塊特殊或字元特殊設備。

SVR4 流派之系統是以 **/dev/dsk** 目錄存放區塊設備，以 **/dev/rdsk** 存放字元（原始資料，raw）設備。同一設備能在提供逆向支援或某些特定功能下，以不同檔名被呼叫。

硬碟的資料區是由許多磁盤表面上多同心圓的磁軌 (tracks) 形成的磁柱 (cylinders) 所組成。資料則是記錄於磁軌的各磁區 (sectors)上。

UNIX 檔案系統通常都由許多個別的檔案系統組成，每一檔案系統都有其本身的根目錄與設備檔。檔案系統係存放於各自獨立的磁碟分割區。將硬碟空間作切

割，能限制破碎區塊資料的量，並確保各分割區的資料不至影響到其他區域。而且有一個特定的設備檔能代表整個硬碟。

每個檔案系統都是由開機區塊、超級區塊、inode 與資料區塊所組成。開機區塊中包含開機所需之資訊與分割區表。超級區塊中則包含檔案系統中全區域性的資訊，如可使用的空 inode 與資料區塊詳細列表。在記憶體中的超級區塊資料會週期性的由 sync 寫回硬碟。許多系統會複製超級區塊資料到硬碟的不同區域，以應不時之需。

Inode 中包含除了檔名以外的所有檔案屬性資料。它建有一十五個位址之陣列，前十二個存放的是檔案的前十二個資料區塊位址。剩下的三個則是形成連結串列(linked list)來存放檔案剩下資料區塊數。如同超級區塊一樣，在記憶體中的 inode 資料會定時寫回硬碟，以保持兩者的同步。

目錄檔中只有記錄 inode 數字和該 inode 有關並存放於該目錄的檔案名稱。當檔案被鏈結時，inode 的鏈結數會隨著遞增。而當 inode 鏈結數歸零時，代表此檔案已被刪除。

每個 UNIX 系統都有 *swap* 與 *root*檔案系統。今日大多數系統都採用能複製保存超級區塊資料、支援符號連結與能管理硬碟空間配額的 *ufs* 檔案系統。Linux 則是採用 *ext2* 檔案系統。此外尚有許多不同的檔案類型，如 CD-ROM 的*hsfs* 或 *iso9660*、DOS 磁碟的 *pcfs*, *vfat* 或 *msdos* 及存放process的虛擬檔案系統*proc* 或 *procfs*。

在某些系統上，fdisk 指令用以製作、刪除與啟動分割區。Linux 系統還允許製作能包含許多邏輯分割區的延伸分割區。mkfs 指令則是用於製作檔案系統，通常是與前端的工具程式一起使用。

在檔案系統未掛上之前，root 檔案系統都看不到此檔案系統。mount 能從 **/etc/fstab**（Solaris 中為 **/etc/vfstab**）取得掛上檔案系統的相關參數資訊。而當檔案系統中有檔案被打開或使用者正處於其下目錄之間，將無法卸下此檔案系統。除此之外，檔案系統都能正常的掛上、卸下。

當系統關機時，如果未將記憶體中的超級區塊與 inode 資料寫回硬碟，檔案系統就可能受到損壞。fsck 會檢查檔案系統的一致性，其通常於卸下狀態時執行。它會嘗試將錯誤檔案資料刪除或將未連結檔移至 **/lost+found** 目錄下，以修復檔案系統。而 root 檔案系統則需於單人使用模式下，才能進行檔案系統檢查。

shutdown 指令在系統關機前，會使用 sync 將記憶體的超級區塊資料寫回檔案系統。但在某些狀況下，硬碟中的資料會比記憶體中的新、正確，此時就不宜進行“ syncing ”（同步） 動作。

自我測驗

21.1 請問 System V 與 Linux 系統通用的軟碟機設備名稱為何？

21.2 請問您用來開機的 3.5 吋 1.44 MB 軟碟機的設備名稱為何？

21.3 請解釋 Linux 系統上之延伸分割區的概念。

21.4 inode 中唯一沒記錄的檔案屬性為何？那請問遺漏的這項屬性又是存在哪裡？

21.5 請問檔案刪除後，inode 會發生什麼變化？

21.6 請問可資使用空的inode 與資料區塊列表是存在何處？

21.7 請問 SVR4 與 Linux 系統將系統核心程式檔存放於哪個目錄？

21.8 請問您要如何找到檔案的 inode 數字？

21.9 請問於 IDE 與 SCSI 硬碟中，您最多能建立多少分割區？

21.10 請問要用哪一個前端工具程式以製作檔案系統？

21.11 請問於哪種狀況下，不能卸下檔案系統？

21.12 請問 fsck 一般對未連結檔都是如何處置？

21.13 請問您要如何對 root 檔案系統作 fsck？

練　習

21.1 請問主要與次要設備數字代表意義為何？

21.2 假如某目錄之資料有可能填滿整個硬碟空間，請問您要如何調整此檔案系統的組織架構？

21.3 請問 Solaris 系統上之設備檔 **/dev/rdsk/c0t3d0s2** 代表哪個分割區，它又有何特殊意義？

21.4 請問 Linux 系統代表整個第一部 IDE 硬碟機的設備檔名稱為何？假如您有一包含兩個邏輯分割區的延伸分割區，該邏輯分割區的設備檔名稱為何？

21.5 請問位於同一主機上的兩個檔案能有一樣的 inode 數字嗎？

21.6 請問目錄資料存放於檔案系統何處？

21.7 請問目錄檔與 inode 中共同的元件有哪些？

21.8 就您所知 UNIX 檔案沒有檔案終結 end-of-file 標記，系統核心如何於顯示該檔時得知其檔案終結點？

21.9 請問 sync在超級區塊與 inodes 之間所扮演的角色為何？為何在系統關機前，慣例上會執行兩個 sync 指令？

21.10 請問分割區表存在何處？在 Linux 系統中它又被稱為什麼？

21.11 為何您需要建立 swap 檔案系統？

21.12　請指出 *proc* 或 *procfs* 檔案系統的三個重要功能。

21.13　請問 SVR4 所使用的標準檔案系統為何？此檔案系統有哪些特殊功能？

21.14　請問 CD-ROM所使用的檔案系統為何？Rock Ridge 此處又扮演什麼角色？

21.15　請問 UNIX 系統提供什麼功能，以修復毀損的超級區塊資料？

21.16　假如要掛上的目錄不是空目錄，您依舊能將檔案系統由此掛上嗎？

21.17　請問 `mount` 於系統啟動時，會讀取以掛上檔案系統的設定檔為何？

21.18　假如您的 Linux 機器上的 DOS 或 Windows 分割區中大多數檔名出現（波浪符號，tilde）的符號，請問您犯了什麼錯誤？要如何才能看到原有的長檔名？

21.19　當檔案系統用唯讀模式被掛載，您需要先卸下此檔案系統才能更動為讀寫模式嗎？

21.20　請問於記憶體中的超級區塊資料是否永遠比硬碟中的資料新？