LINUX磁碟與檔案系統管理

陳建良

內容

- 認識 EXT2 檔案系統
- 檔案系統的簡單操作
- 磁碟的分割、格式化、檢驗與掛載
- 設定開機掛載
- 記憶體置換空間(swap)之建置
- 檔案系統的特殊觀察與操作

認識 EXT2 檔案系統

- 相硬碟組成與分割的複習
  - 圓形的磁碟盤(主要記錄資料的部分)；
  - 機械手臂，與在機械手臂上的磁碟讀取頭(可讀寫磁碟盤上的資料)；
  - 主軸馬達，可以轉動磁碟盤，讓機械手臂的讀取頭在磁碟盤上讀寫資料。
  - 磁區(Sector)為最小的物理儲存單位，每個磁區為 512 bytes；
  - 將磁區組成一個圓，那就是磁柱(Cylinder)，磁柱是分割槽(partition)的最小單位；
  - 第一個磁區最重要，裡面有：(1)主要開機區(Master boot record, MBR)及分割表(partition table)，其中 MBR 佔有 446 bytes，而 partition table 則佔有 64 bytes。

分割定義複習

- 主要分割與延伸分割最多可以有四筆(硬碟的限制)
- 延伸分割最多只能有一個(作業系統的限制)
- 邏輯分割是由延伸分割持續切割出來的分割槽；
- 能夠被格式化後，作為資料存取的分割槽為主要分割與邏輯分割。延伸分割無法格式化；
- 邏輯分割的數量依作業系統而不同，在Linux系統中，IDE硬碟最多有59個邏輯分割(5號到63號)， SATA硬碟則有11個邏輯分割(5號到15號)。

檔案系統格式(filesystem)

- 我們都知道磁碟分割完畢後還需要進行格式化(format)
- 為什麼需要進行『格式化』呢？
  - 因為每種作業系統所設定的檔案屬性/權限並不相同， 為了存放這些檔案所需的資料，因此就需要將分割槽進行格式化
- 傳統的磁碟與檔案系統之應用中，一個分割槽就是只能夠被格式化成為一個檔案系統，所以我們可以說一個 filesystem 就是一個 partition

檔案系統格式(filesystem)

- 由於新技術的利用，例如我們常聽到的LVM與軟體磁碟陣列(software raid)
  - LVM技術可以將一個分割槽格式化為多個檔案系統
  - LVM, RAID能夠將多個分割槽合成一個檔案系統
- 因此我們可以稱呼一個可被掛載的資料為一個檔案系統而不是一個分割槽

檔案系統格式(filesystem)

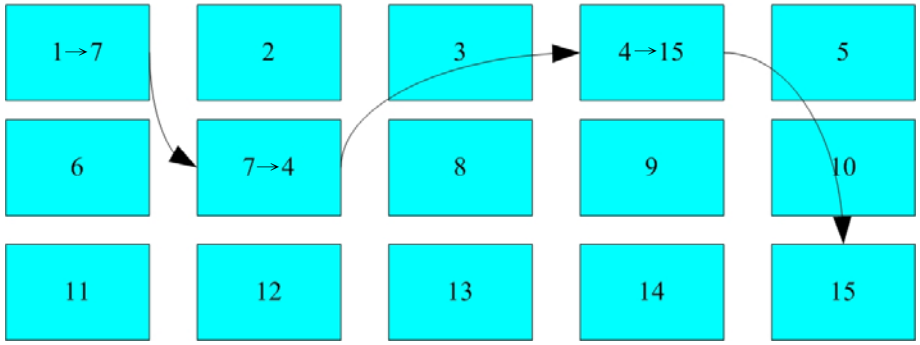
- 檔案資料除了檔案實際內容外， 通常含有非常多的屬性，例如 Linux 作業系統的檔案權限(rwx)與檔案屬性(擁有者、群組、時間參數等)
- 通常會將這兩部份的資料分別存放在不同的區塊，權限與屬性放置到 inode 中，至於實際資料則放置到 data block 區塊中
- 另外還有一個超級區塊 (superblock) 會記錄整個檔案系統的整體資訊

檔案系統格式(filesystem)

- superblock：記錄此 filesystem 的整體資訊，包括 inode/block的總量、使用量、剩餘量， 以及檔案系統的格式與相關資訊等；
- inode：記錄檔案的屬性，一個檔案佔用一個inode，同時記錄此檔案的資料所在的 block 號碼；
- block：實際記錄檔案的內容，若檔案太大時，會佔用多個 block 。

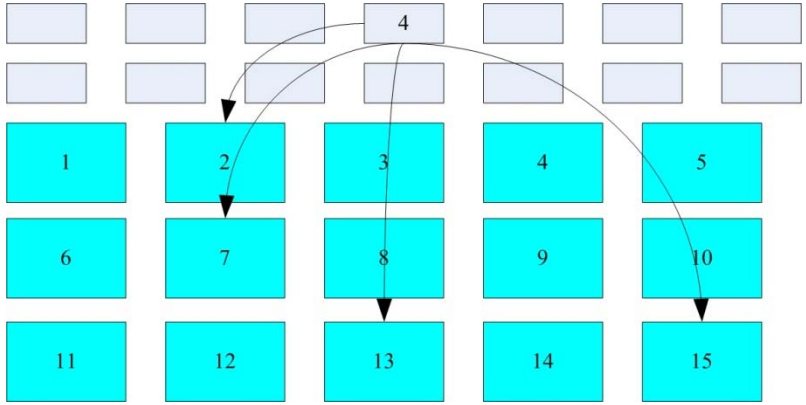
認識 EXT2 檔案系統

檔案系統特性



FAT檔案系統資料存取示意圖

- 檔案系統沒有辦法一口氣就知道四個 block 的號碼，他得要一個一個的將 block 讀出後，才會知道下一個 block 在何處。



inode/block 資料存取示意圖

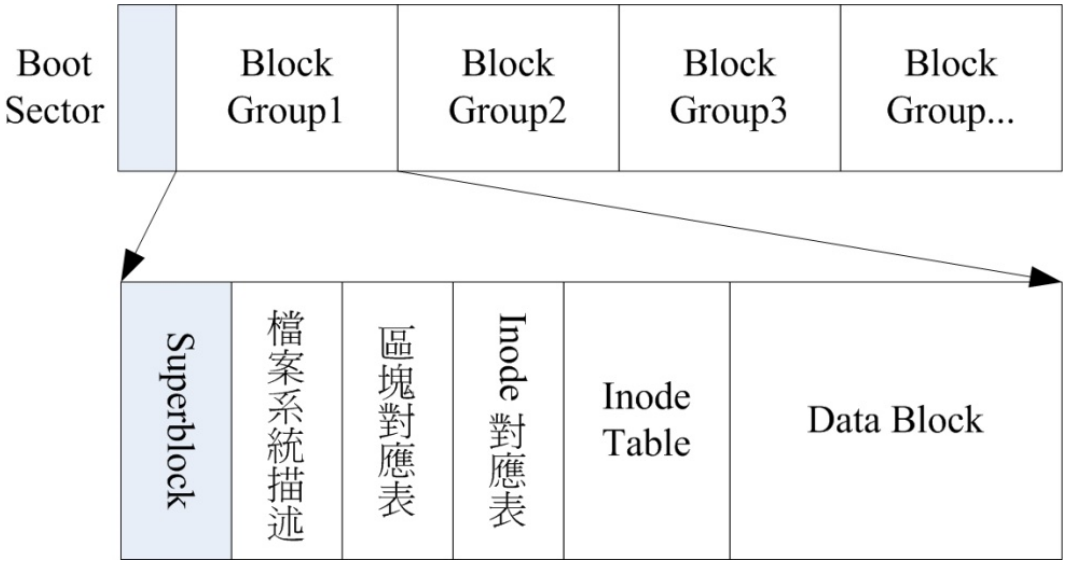
- 假設某一個檔案的屬性與權限資料是放置到 inode 4 號，而這個 inode 記錄了檔案資料的實際放置點為 2, 7, 13, 15 這四個 block 號碼，此時我們的作業系統就能夠據此來排列磁碟的讀取順序，可以一口氣將四個 block 內容讀出來。這種資料存取的方法我們稱為索引式檔案系統 (indexed allocation)

磁碟重組

- 需要磁碟重組的原因就是檔案寫入的 block 太過於離散了，此時檔案讀取的效能將會變的很差所致
- 可以透過磁碟重組將同一個檔案所屬的 blocks 彙整在一起，這樣資料的讀取會比較容易
- FAT 的檔案系統需要三不五時的磁碟重組一下，那麼 Ext2 是否需要磁碟重整呢？

- 如果檔案系統高達數百GB時， 那麼將所有的 inode 與 block 通通放置在一起將是很不智的決定，因為 inode 與 block 的數量太龐大，不容易管理。
- 因此 Ext2 檔案系統在格式化的時候基本上是區分為多個區塊群組 (block group) 的，每個區塊群組都有獨立的 inode/block/superblock 系統。

Linux 的 EXT2 檔案系統(inode)



ext2檔案系統示意圖

Data Block (資料區塊)

- data block 是用來放置檔案內容資料地方，在 Ext2 檔案系統中所支援的 block 大小有 1K, 2K 及 4K 三種而已。在格式化時 block 的大小就固定了，且每個 block 都有編號，以方便 inode 的記錄啦。 不過要注意的是，由於 block 大小的差異，會導致該檔案系統能夠支援的最大磁碟容量與最大單一檔案容量並不相同。 因為 block 大小而產生的 Ext2 檔案系統限制如下：

Block 大小	1KB	2KB	4KB
最大單一檔案限制	16GB	256GB	2TB
最大檔案系統總容量	2TB	8TB	16TB

- 除此之外 Ext2 檔案系統的 block 還有什麼限制呢？有的！基本限制如下：
  - 原則上，block 的大小與數量在格式化完就不能夠再改變了(除非重新格式化)；
  - 每個 block 內最多只能夠放置一個檔案的資料；
  - 承上，如果檔案大於 block 的大小，則一個檔案會佔用多個 block 數量；
  - 承上，若檔案小於 block ，則該 block 的剩餘容量就不能夠再被使用了(磁碟空間會浪費)。
- 如上第四點所說，由於每個 block 僅能容納一個檔案的資料而已，因此如果你的檔案都非常小，但是你的 block 在格式化時卻選用最大的 4K 時，可能會產生一些容量的浪費喔！我們以底下的一個簡單例題來算一下空間的浪費吧！

- 例題：假設你的Ext2檔案系統使用 4K block ，而該檔案系統中有 10000 個小檔案，每個檔案大小均為 50bytes，請問此時你的磁碟浪費多少容量？
- 答：由於 Ext2 檔案系統中一個 block 僅能容納一個檔案，因此每個 block 會浪費『 4096 - 50 = 4046 (byte)』， 系統中總共有一萬個小檔案，所有檔案容量為：50 (bytes) x 10000 = 488.3Kbytes，但此時浪費的容量為：『 4046 (bytes) x 10000 = 38.6MBytes 』。想一想，不到 1MB 的總檔案容量卻浪費將近 40MB 的容量，且檔案越多將造成越多的磁碟容量浪費。

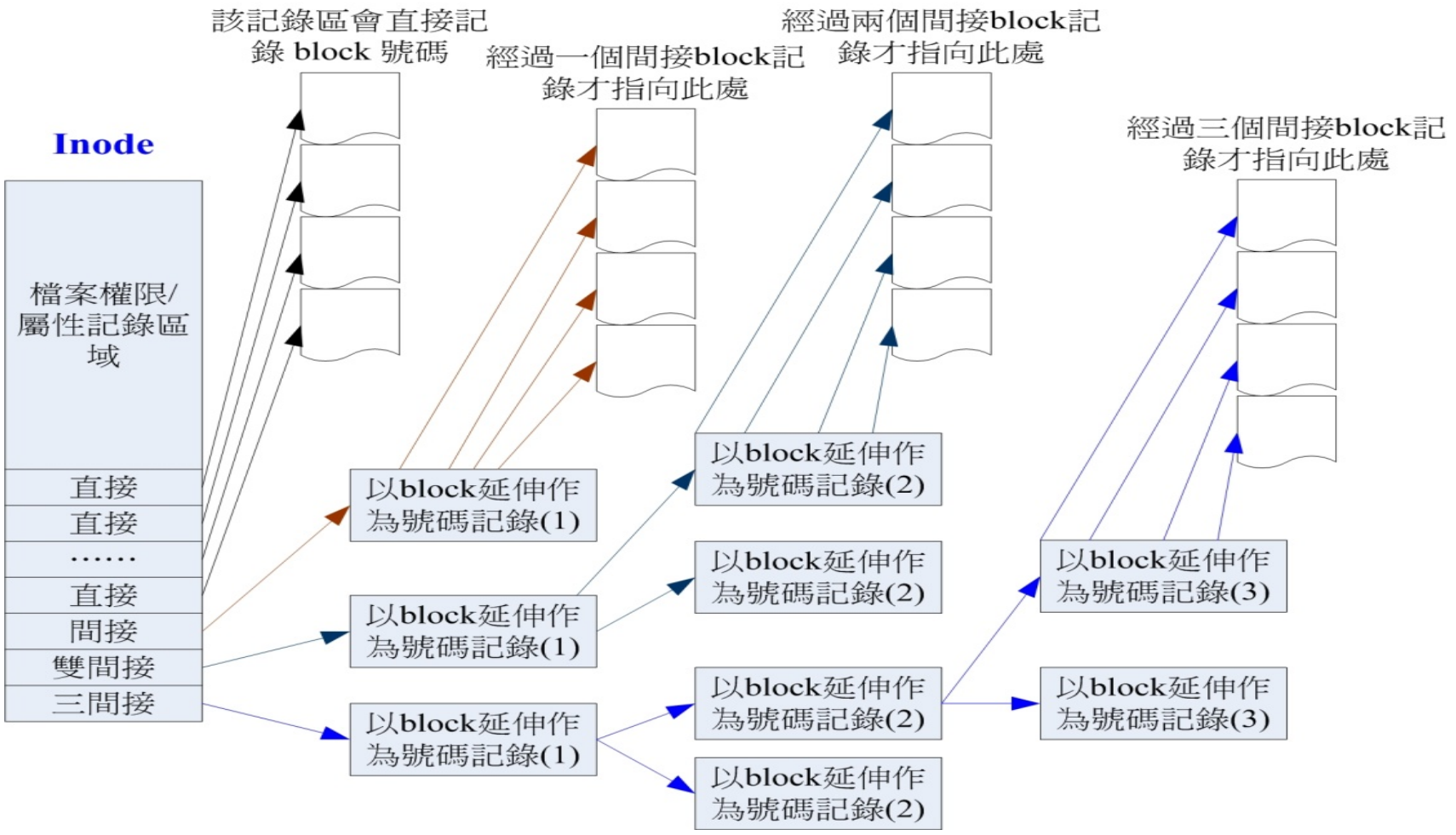
inode table (inode 表格)

- inode 記錄的檔案資料至少有底下這些：
  - 該檔案的存取模式(read/write/excute)；
  - 該檔案的擁有者與群組(owner/group)；
  - 該檔案的容量；
  - 該檔案建立或狀態改變的時間(ctime)；
  - 最近一次的讀取時間(atime)；
  - 最近修改的時間(mtime)；
  - 定義檔案特性的旗標(flag)，如 SetUID...；
  - 該檔案真正內容的指向 (pointer)；

inode table (inode 表格)

- inode 的數量與大小也是在格式化時就已經固定了，除此之外 inode 還有些什麼特色呢？
  - 每個 inode 大小均固定為 128 bytes；
  - 每個檔案都僅會佔用一個 inode 而已；
  - 承上，因此檔案系統能夠建立的檔案數量與 inode 的數量有關；
  - 系統讀取檔案時需要先找到 inode，並分析 inode 所記錄的權限與使用者是否符合，若符合才能夠開始實際讀取 block 的內容。

- inode 要記錄的資料非常多，但偏偏又只有 128bytes 而已， 而 inode 記錄一個 block 號碼要花掉 4byte ，假設我一個檔案有 400MB 且每個 block 為 4K 時， 那麼至少也要十萬筆 block 號碼的記錄呢！
- inode 哪有這麼多可記錄的資訊？為此我們的系統很聰明的將 inode 記錄 block 號碼的區域定義為
  - 12個直接
  - 一個間接
  - 一個雙間接
  - 與一個三間接記錄區


該記錄區會直接記錄 block 號碼
經過一個間接block記錄才指向此處
經過兩個間接block記錄才指向此處
經過三個間接block記錄才指向此處
Inode
檔案權限/屬性記錄區域
直接
直接
.....
直接
間接
雙間接
三間接
以block延伸作為號碼記錄(1)
以block延伸作為號碼記錄(1)
以block延伸作為號碼記錄(1)
以block延伸作為號碼記錄(2)
以block延伸作為號碼記錄(2)
以block延伸作為號碼記錄(2)
以block延伸作為號碼記錄(2)
以block延伸作為號碼記錄(3)
以block延伸作為號碼記錄(3)

- 這樣子 inode 能夠指定多少個 block 呢？我們以較小的 1K block 來說明好了，可以指定的情況如下：
  - 12 個直接指向： 12*1K=12K
    由於是直接指向，所以總共可記錄 12 筆記錄，因此總額大小為如上所示；
  - 間接： 256*1K=256K
    每筆 block 號碼的記錄會花去 4bytes，因此 1K 的大小能夠記錄 256 筆記錄，因此一個間接可以記錄的檔案大小如上；
  - 雙間接： $256*256*1K=256^2K$
    第一層 block 會指定 256 個第二層，每個第二層可以指定 256 個號碼，因此總額大小如上；
  - 三間接： $256*256*256*1K=256^3K$
    第一層 block 會指定 256 個第二層，每個第二層可以指定 256 個第三層，每個第三層可以指定 256 個號碼，因此總額大小如上；
- 總額：將直接、間接、雙間接、三間接加總，得到 12 + 256 + 256*256 + 256*256*256 (K) = 16GB

認識 EXT2 檔案系統

- Superblock (超級區塊)
  - block 與 inode 的總量；
  - 未使用與已使用的 inode / block 數量；
  - block 與 inode 的大小 (block 為 1, 2, 4K，inode 為 128 bytes)；
  - filesystem 的掛載時間、最近一次寫入資料的時間、最近一次檢驗磁碟 (fsck) 的時間等檔案系統的相關資訊；
  - 一個 valid bit 數值，若此檔案系統已被掛載，則 valid bit 為 0 ，若未被掛載，則 valid bit 為 1 。
- Superblock 是非常重要的，因為我們這個檔案系統的基本資訊都寫在這裡，因此，如果 superblock 死掉了， 你的檔案系統可能就需要花費很多時間去挽救啦！一般來說，superblock 的大小為 1024bytes。相關的 superblock 訊息我們等一下會以 dumpe2fs 指令來呼叫出來觀察喔！

```
[root@www ~]# dumpe2fs [-bh] 裝置檔名
選項與參數：
-b ：列出保留為壞軌的部分(一般用不到吧！？)
-h ：僅列出 superblock 的資料，不會列出其他的區段內容！

範例：找出我的根目錄磁碟檔名，並觀察檔案系統的相關資訊
[root@www ~]# df   <==這個指令可以叫出目前掛載的裝置
Filesystem    1K-blocks      Used Available Use% Mounted on
/dev/hdc2       9920624   3822848   5585708  41% /         <==就是這個光！
/dev/hdc3       4956316    141376   4559108   4% /home
/dev/hdc1        101086     11126     84741  12% /boot
tmpfs            371332         0    371332   0% /dev/shm

[root@www ~]# dumpe2fs /dev/hdc2
dumpe2fs 1.39 (29-May-2006)
Filesystem volume name:   /1                  <==這個是檔案系統的名稱(Label)
Filesystem features:      has_journal ext_attr resize_inode dir_index
  filetype needs_recovery sparse_super large_file
Default mount options:    user_xattr acl <==預設掛載的參數
Filesystem state:         clean              <==這個檔案系統是沒問題的(clean)
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              2560864            <==inode的總數
Block count:              2560359            <==block的總數
Free blocks:              1524760            <==還有多少個 block 可用
Free inodes:              2411225            <==還有多少個 inode 可用
First block:              0
Block size:               4096               <==每個 block 的大小啦！
Filesystem created:       Fri Sep  5 01:49:20 2008
Last mount time:          Mon Sep 22 12:09:30 2008
Last write time:          Mon Sep 22 12:09:30 2008
```

```
Inode size:               128           <==每個 inode 的大小
Journal inode:              8             <==底下這三個與下一小節有關
Journal backup:             inode blocks
Journal size:               128M

Group 0: (Blocks 0-32767) <==第一個 data group 內容, 包含 block 的啟始/結束號碼
  Primary superblock at 0, Group descriptors at 1-1  <==超級區塊在 0 號 block
  Reserved GDT blocks at 2-626
  Block bitmap at 627 (+627), Inode bitmap at 628 (+628)
  Inode table at 629-1641 (+629)                     <==inode table 所在的 block
  0 free blocks, 32405 free inodes, 2 directories    <==所有 block 都用完了！
  Free blocks:
  Free inodes: 12-32416                              <==剩餘未使用的 inode 號碼
Group 1: (Blocks 32768-65535)
```

- 由於 (1)一個 inode 佔用 128 bytes ，(2)總共有 1641 - 629 + 1(629本身) = 1013 個 block 花在 inode table 上， (3)每個 block 的大小為 4096 bytes(4K)。由這些數據可以算出 inode 的數量共有 1013 * 4096 / 128 = 32416 個 inode 啦！

- Filesystem Description (檔案系統描述說明)
  - 這個區段可以描述每個 block group 的開始與結束的 block 號碼，以及說明每個區段 (superblock, bitmap, inodemap, data block) 分別介於哪一個 block 號碼之間。這部份也能夠用 dumpe2fs 來觀察的。
- block bitmap (區塊對照表)
  - 如果你想要新增檔案時總會用到 block 吧！那你要使用哪個 block 來記錄呢？當然是選擇『空的 block 』來記錄新檔案的資料囉。那你怎麼知道哪個 block 是空的？這就得要透過 block bitmap 的輔助了。從 block bitmap 當中可以知道哪些 block 是空的，因此我們的系統就能夠很快速的找到可使用的空間來處置檔案囉。
- inode bitmap (inode 對照表)
  - 這個其實與 block bitmap 是類似的功能，只是 block bitmap 記錄的是使用與未使用的 block 號碼， 至於 inode bitmap 則是記錄使用與未使用的 inode 號碼囉！

與目錄樹的關係

- 在 Linux 下的 ext2 檔案系統建立一個目錄時， ext2 會分配一個 inode 與至少一塊 block 給該目錄。其中，inode 記錄該目錄的相關權限與屬性，並可記錄分配到的那塊 block 號碼； 而 block 則是記錄在這個目錄下的檔名與該檔名佔用的 inode 號碼資料。也就是說目錄所佔用的 block 內容在記錄如下的資訊：

Inode number	檔名
654683	anaconda-ks.cfg
648322	install.log
648323	install.log.syslog
…	…

記載於目錄所屬的 block 內的檔名與 inode 號碼對應示意圖

```
[root@www ~]# ls -li
total 92
654683 -rw------- 1 root root  1474 Sep  4 18:27 anaconda-ks.cfg
648322 -rw-r--r-- 1 root root 42304 Sep  4 18:26 install.log
648323 -rw-r--r-- 1 root root  5661 Sep  4 18:25 install.log.syslog
```

檔案：

- 當我們在 Linux 下的 ext2 建立一個一般檔案時， ext2 會分配一個 inode 與相對於該檔案大小的 block 數量給該檔案。例如：假設我的一個 block 為 4 Kbytes ，而我要建立一個 100 KBytes 的檔案，那麼 linux 將分配一個 inode 與 25 個 block 來儲存該檔案！但同時請注意，由於 inode 僅有 12 個直接指向，因此還要多一個 block 來作為區塊號碼的記錄！

- 目錄樹讀取：
  - 我們很清楚的知道 inode 本身並不記錄檔名，檔名的記錄是在目錄的 block 當中。
  - 因為檔名是記錄在目錄的 block 當中， 因此當我們要讀取某個檔案時，就務必會經過目錄的 inode 與 block ，然後才能夠找到那個待讀取檔案的 inode 號碼， 最終才會讀到正確的檔案的 block 內的資料。
  - 由於目錄樹是由根目錄開始讀起，因此系統透過掛載的資訊可以找到掛載點的 inode 號碼(通常一個 filesystem 的最頂層 inode 號碼會由 2 號開始喔！)，此時就能夠得到根目錄的 inode 內容，並依據該 inode 讀取根目錄的 block 內的檔名資料，再一層一層的往下讀到正確的檔名。

- 舉例來說，如果我想要讀取 /etc/passwd 這個檔案時，系統是如何讀取的呢？

```
[root@www ~]# ll -di / /etc /etc/passwd
      2 drwxr-xr-x  23 root root  4096 Sep 22 12:09 /
1912545 drwxr-xr-x 105 root root 12288 Oct 14 04:02 /etc
1914888 -rw-r--r--   1 root root  1945 Sep 29 02:21 /etc/passwd
```

- / 的 inode：
  透過掛載點的資訊找到 /dev/hdc2 的 inode 號碼為 2 的根目錄 inode，且 inode 規範的權限讓我們可以讀取該 block 的內容(有 r 與 x) ；
- / 的 block：
  經過上個步驟取得 block 的號碼，並找到該內容有 etc/ 目錄的 inode 號碼 (1912545)；
- etc/ 的 inode：
  讀取 1912545 號 inode 得知 vbird 具有 r 與 x 的權限，因此可以讀取 etc/ 的 block 內容；
- etc/ 的 block：
  經過上個步驟取得 block 號碼，並找到該內容有 passwd 檔案的 inode 號碼 (1914888)；
- passwd 的 inode：
  讀取 1914888 號 inode 得知 vbird 具有 r 的權限，因此可以讀取 passwd 的 block 內容；
- passwd 的 block：
  最後將該 block 內容的資料讀出來。

- **filesystem 大小與磁碟讀取效能：**
- 當你的一個檔案系統規劃的很大時，例如 100GB 這麼大時， 由於硬碟上面的資料總是來來去去的，所以，整個檔案系統上面的檔案通常無法連續寫在一起(block 號碼不會連續的意思)， 而是填入式的將資料填入沒有被使用的 block 當中。如果檔案寫入的 block 真的分的很散， 此時就會有所謂的檔案資料離散的問題發生了。
- 如前所述，雖然我們的 ext2 在 inode 處已經將該檔案所記錄的 block 號碼都記上了， 所以資料可以一次性讀取，但是如果檔案真的太過離散，確實還是會發生讀取效率低落的問題。 因為磁碟讀取頭還是得要在整個檔案系統中來來去去的頻繁讀取！ 果真如此，那麼可以將整個 filesystme 內的資料全部複製出來，將該 filesystem 重新格式化， 再將資料給他複製回去即可解決這個問題。
- 此外，如果 filesystem 真的太大了，那麼當一個檔案分別記錄在這個檔案系統的最前面與最後面的 block 號碼中， 此時會造成硬碟的機械手臂移動幅度過大，也會造成資料讀取效能的低落。而且讀取頭在搜尋整個 filesystem 時， 也會花費比較多的時間去搜尋！因此， partition 的規劃並不是越大越好， 而是真的要針對您的主機用途來進行規劃才行！

- 如果是新建一個檔案或目錄時，我們的 Ext2 是如何處理的呢？ 這個時候就得要 block bitmap 及 inode bitmap 的幫忙了！假設我們想要新增一個檔案，此時檔案系統的行為是：
  - 1.先確定使用者對於欲新增檔案的目錄是否具有 w 與 x 的權限，若有的話才能新增；
  - 2根據 inode bitmap 找到沒有使用的 inode 號碼，並將新檔案的權限/屬性寫入；
  - 3根據 block bitmap 找到沒有使用中的 block 號碼，並將實際的資料寫入 block 中，且更新 inode 的 block 指向資料；
  - 4.將剛剛寫入的 inode 與 block 資料同步更新 inode bitmap 與 block bitmap，並更新 superblock 的內容。

資料的不一致 (Inconsistent) 狀態

- 例如你的檔案在寫入檔案系統時，因為不知名原因導致系統中斷(例如突然的停電啊、 系統核心發生錯誤啊 ~ 等等的怪事發生時)，所以寫入的資料僅有 inode table 及 data block 而已，最後一個同步更新中介資料的步驟並沒有做完，此時就會發生 metadata 的內容與實際資料存放區產生不一致 (Inconsistent) 的情況了。
- 在早期的 Ext2 檔案系統中，如果發生這個問題， 那麼系統在重新開機的時候，就會藉由 Superblock 當中記錄的 valid bit (是否有掛載) 與 filesystem state (clean 與否) 等狀態來判斷是否強制進行資料一致性的檢查！若有需要檢查時則以 e2fsck 這支程式來進行的

日誌式檔案系統 (Journaling Filesystem)

- 避免上述提到的檔案系統不一致的情況發生，因此在我們的 filesystem 當中規劃出一個區塊，該區塊專門在記錄寫入或修訂檔案時的步驟， 那不就可以簡化一致性檢查的步驟了？也就是說：
- 1.預備：當系統要寫入一個檔案時，會先在日誌記錄區塊中記錄某個檔案準備要寫入的資訊；
- 2.實際寫入：開始寫入檔案的權限與資料；開始更新 metadata 的資料；
- 3.結束：完成資料與 metadata 的更新後，在日誌記錄區塊當中完成該檔案的記錄。
- 利用dumpe2fs 輸出的訊息，可以發現 superblock 裡面含有底下這樣的資訊：

```
Journal inode:            8
Journal backup:           inode blocks
Journal size:             128M
```

Linux 檔案系統的運作

- 我們知道 所有的資料都得要載入到記憶體後 CPU 才能夠對該資料進行處理。想一想，如果你常常編輯一個好大的檔案，在編輯的過程中又頻繁的要系統來寫入到磁碟中
- 為了解決這個效率的問題，因此我們的 Linux 使用的方式是透過一個稱為非同步處理 (asynchronously) 的方式。所謂的非同步處理是這樣的：
  - 當系統載入一個檔案到記憶體後，如果該檔案沒有被更動過，則在記憶體區段的檔案資料會被設定為乾淨(clean)的。 但如果記憶體中的檔案資料被更改過了(例如你用 nano 去編輯過這個檔案)，此時該記憶體中的資料會被設定為髒的 (Dirty)。此時所有的動作都還在記憶體中執行，並沒有寫入到磁碟中！ 系統會不定時的將記憶體中設定為『Dirty』的資料寫回磁碟，以保持磁碟與記憶體資料的一致性。

Linux 檔案系統的運作

- 系統會將常用的檔案資料放置到主記憶體的緩衝區，以加速檔案系統的讀/寫；
- 因此 Linux 的實體記憶體最後都會被用光！這是正常的情況！可加速系統效能；
- 你可以手動使用 sync 來強迫記憶體中設定為 Dirty 的檔案回寫到磁碟中；
- 若正常關機時，關機指令會主動呼叫 sync 來將記憶體的資料回寫入磁碟內；
- 但若不正常關機(如跳電、當機或其他不明原因)，由於資料尚未回寫到磁碟內，因此重新開機後可能會花很多時間在進行磁碟檢驗，甚至可能導致檔案系統的損毀(非磁碟損毀)。

掛載點的意義 (Mount Point)

- 每個 filesystem 都有獨立的 inode / block / superblock 等資訊，這個檔案系統要能夠連結到目錄樹才能被我們使用。 將檔案系統與目錄樹結合的動作我們稱為『掛載』。
- 假設有三個掛載點，分別是 /, /boot, /home 三個(裝置檔名為 /dev/hdc2, /dev/hdc1, /dev/hdc3)，觀察這三個目錄的 inode 號碼時，我們可以發現如下的情況：

```
[root@www ~]# ls -lid / /boot /home
2 drwxr-xr-x 23 root root 4096 Sep 22 12:09 /
2 drwxr-xr-x  4 root root 1024 Sep  4 18:06 /boot
2 drwxr-xr-x  6 root root 4096 Sep 29 02:21 /home
```

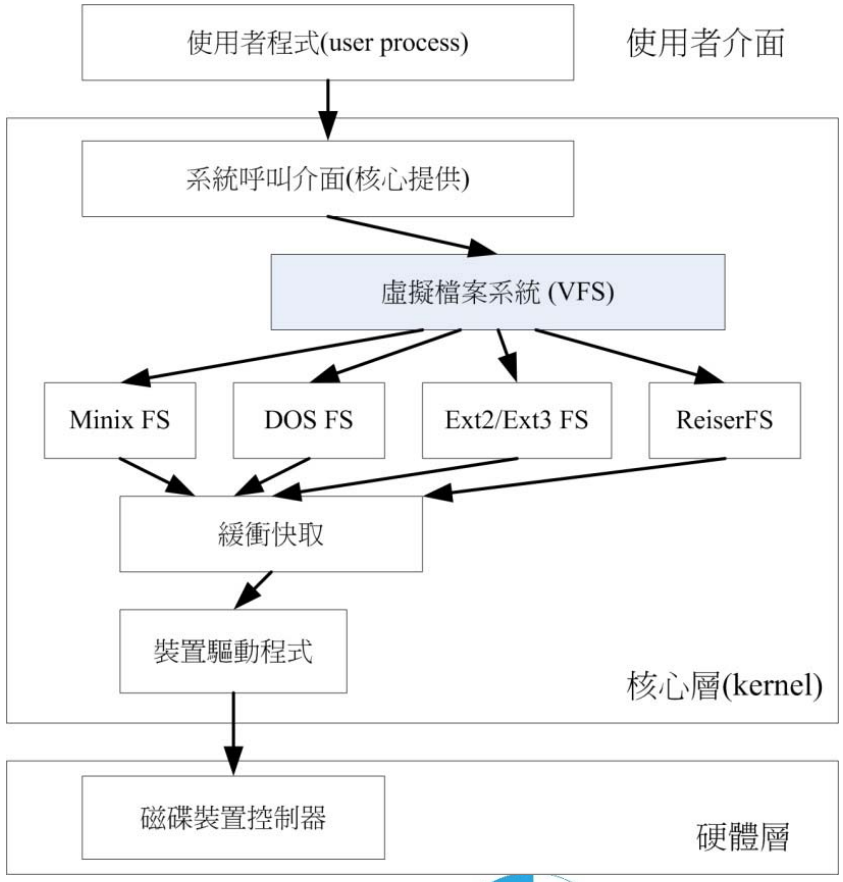
掛載點的意義 (Mount Point)

- 如果使用檔案系統的觀點來看，同一個 filesystem 的某個 inode 只會對應到一個檔案內容而已(因為一個檔案佔用一個 inode 之故)， 因此我們可以透過判斷 inode 號碼來確認不同檔名是否為相同的檔案！所以可以這樣看：

```
[root@www ~]# ls -ild /  /.  /..
2 drwxr-xr-x 23 root root 4096 Sep 22 12:09 /
2 drwxr-xr-x 23 root root 4096 Sep 22 12:09 /.
2 drwxr-xr-x 23 root root 4096 Sep 22 12:09 /..
```

其他 Linux 支援的檔案系統與 VFS

- 傳統檔案系統：ext2 / minix / MS-DOS / FAT (用 vfat 模組) / iso9660 (光碟)等等；
- 日誌式檔案系統：ext3 / ReiserFS / Windows' NTFS / IBM's JFS / SGI's XFS
- 網路檔案系統：NFS / SMBFS



VFS 檔案系統的示意圖

Linux VFS (Virtual Filesystem Switch)

- Linux 的系統都是透過一個名為 Virtual Filesystem Switch 的核心功能去讀取 filesystem 的。 也就是說，整個 Linux 認識的 filesystem 其實都是 VFS 在進行管理，我們使用者並不需要知道每個 partition 上頭的 filesystem 是什麼～ VFS 會主動的幫我們做好讀取的動作
- 假設你的 / 使用的是 /dev/hda1 ，用 ext3 ，而 /home 使用 /dev/hda2 ，用 reiserfs ， 那麼你取用 /home/dmtsai/.bashrc 時，有特別指定要用的什麼檔案系統的模組來讀取嗎？ 應該是沒有吧！這個就是 VFS 的功能啦！透過這個 VFS 的功能來管理所有的 filesystem， 省去我們需要自行設定讀取檔案系統的定義

檔案系統的簡單操作

- 磁碟與目錄的容量
  - df：列出檔案系統的整體磁碟使用量；
  - du：評估檔案系統的磁碟使用量(常用在推估目錄所佔容量)

```
[root@www ~]# df [-ahikHTm] [目錄或檔名]
選項與參數：
-a   ：列出所有的檔案系統，包括系統特有的 /proc 等檔案系統；
-k   ：以 KBytes 的容量顯示各檔案系統；
-m   ：以 MBytes 的容量顯示各檔案系統；
-h   ：以人們較易閱讀的 GBytes, MBytes, KBytes 等格式自行顯示；
-H   ：以 M=1000K 取代 M=1024K 的進位方式；
-T   ：連同該 partition 的 filesystem 名稱 (例如 ext3) 也列出；
-i   ：不用硬碟容量，而以 inode 的數量來顯示
```

```
範例一：將系統內所有的 filesystem 列出來！
[root@www ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hdc2              9920624   3823112   5585444  41% /
/dev/hdc3              4956316    141376   4559108   4% /home
/dev/hdc1               101086     11126     84741  12% /boot
tmpfs                   371332         0    371332   0% /dev/shm
# 在 Linux 底下如果 df 沒有加任何選項，那麼預設會將系統內所有的
# (不含特殊記憶體內的檔案系統與 swap) 都以 1 Kbytes 的容量來列出來！
# 至於那個 /dev/shm 是與記憶體有關的掛載，先不要理他！

範例二：將容量結果以易讀的容量格式顯示出來
[root@www ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hdc2             9.5G  3.7G  5.4G  41% /
/dev/hdc3             4.8G  139M  4.4G   4% /home
/dev/hdc1              99M   11M   83M  12% /boot
tmpfs                 363M     0  363M   0% /dev/shm
# 不同於範例一，這裡會以 G/M 等容量格式顯示出來，比較容易看啦！

範例三：將系統內的所有特殊檔案格式及名稱都列出來
[root@www ~]# df -aT
Filesystem    Type 1K-blocks      Used Available Use% Mounted on
/dev/hdc2     ext3   9920624   3823112   5585444  41% /
proc          proc         0         0         0   -  /proc
sysfs         sysfs        0         0         0   -  /sys
devpts        devpts       0         0         0   -  /dev/pts
/dev/hdc3     ext3   4956316    141376   4559108   4% /home
/dev/hdc1     ext3    101086     11126     84741  12% /boot
tmpfs         tmpfs   371332         0    371332   0% /dev/shm
none          binfmt_misc  0         0         0   -  /proc/sys/fs/binfmt_misc
sunrpc        rpc_pipefs   0         0         0   -  /var/lib/nfs/rpc_pipefs
# 系統裡面其實還有很多特殊的檔案系統存在的。那些比較特殊的檔案系統幾乎
# 都是在記憶體當中，例如 /proc 這個掛載點。因此，這些特殊的檔案系統
# 都不會佔據硬碟空間喔！ ^_^
```

```
範例四：將 /etc 底下的可用的磁碟容量以易讀的容量格式顯示
[root@www ~]# df -h /etc
Filesystem            Size  Used Avail Use% Mounted on
/dev/hdc2             9.5G  3.7G  5.4G  41% /
# 這個範例比較有趣一點啦，在 df 後面加上目錄或者是檔案時， df
# 會自動的分析該目錄或檔案所在的 partition ，並將該 partition 的容量顯示出來，
# 所以，您就可以知道某個目錄底下還有多少容量可以使用了！ ^_^

範例五：將目前各個 partition 當中可用的 inode 數量列出
[root@www ~]# df -ih
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hdc2               2.5M    147K    2.3M     6% /
/dev/hdc3               1.3M      46    1.3M     1% /home
/dev/hdc1                26K      34     26K     1% /boot
tmpfs                    91K       1     91K     1% /dev/shm
# 這個範例則主要列出可用的 inode 剩餘量與總容量。分析一下與範例一的關係，
# 你可以清楚的發現到，通常 inode 的數量剩餘都比 block 還要多呢
```

du

[root@www ~]# **du [-ahskm]** 檔案或目錄名稱
選項與參數：
-a ：列出所有的檔案與目錄容量，因為預設僅統計目錄底下的檔案量而已。
-h ：以人們較易讀的容量格式 (G/M) 顯示；
-s ：列出總量而已，而不列出每個各別的目錄佔用容量；
-S ：不包括子目錄下的總計，與 -s 有點差別。
-k ：以 KBytes 列出容量顯示；
-m ：以 MBytes 列出容量顯示；

```
[root@www ~]# du [-ahskm] 檔案或目錄名稱
選項與參數：
-a   ：列出所有的檔案與目錄容量，因為預設僅統計目錄底下的檔案量而已。
-h   ：以人們較易讀的容量格式 (G/M) 顯示；
-s   ：列出總量而已，而不列出每個各別的目錄佔用容量；
-S   ：不包括子目錄下的總計，與 -s 有點差別。
-k   ：以 KBytes 列出容量顯示；
-m   ：以 MBytes 列出容量顯示；

範例一：列出目前目錄下的所有檔案容量
[root@www ~]# du
8       ./test4       <==每個目錄都會列出來
8       ./test2
....中間省略....
12      ./.gconfd     <==包括隱藏檔的目錄
220     .             <==這個目錄(.)所佔用的總量
# 直接輸入 du 沒有加任何選項時，則 du 會分析『目前所在目錄』
# 的檔案與目錄所佔用的硬碟空間。但是，實際顯示時，僅會顯示目錄容量(不含檔案)，
# 因此 . 目錄有很多檔案沒有被列出來，所以全部的目錄相加不會等於 . 的容量喔！
# 此外，輸出的數值資料為 1K 大小的容量單位。

範例二：同範例一，但是將檔案的容量也列出來
[root@www ~]# du -a
12      ./install.log.syslog    <==有檔案的列表了
8       ./.bash_logout
8       ./test4
8       ./test2
....中間省略....
12      ./.gconfd
220     .

範例三：檢查根目錄底下每個目錄所佔用的容量
[root@www ~]# du -sm /*
7       /bin
6       /boot
.....中間省略....
0       /proc
.....中間省略....
1       /tmp
3859    /usr        <==系統初期最大就是他了啦！
77      /var
# 這是個很常被使用的功能～利用萬用字元 * 來代表每個目錄，
# 如果想要檢查某個目錄下，哪個次目錄佔用最大的容量，可以用這個方法找出來
# 值得注意的是，如果剛剛安裝好 Linux 時，那麼整個系統容量最大的應該是 /usr
```

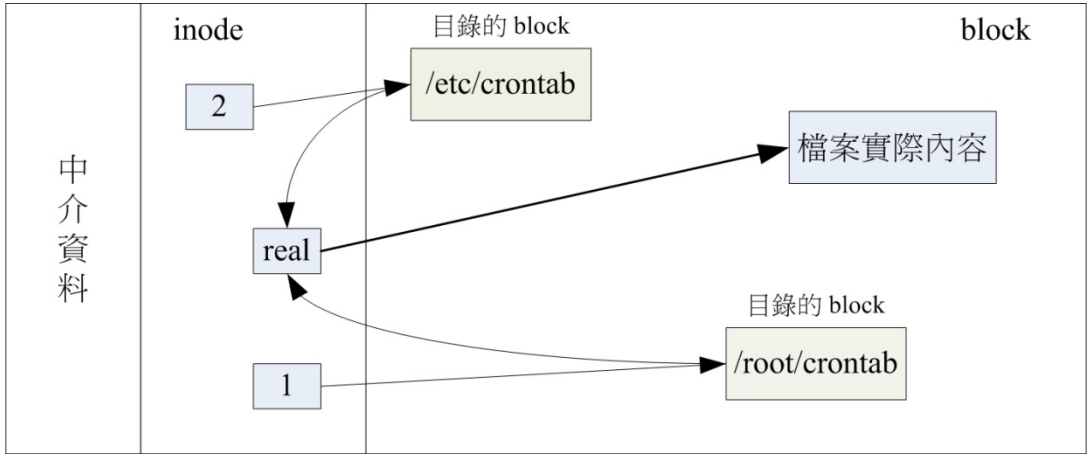
檔案系統的簡單操作

- 在前一小節當中，我們知道幾件重要的資訊，包括：
  - 每個檔案都會佔用一個 inode ，檔案內容由 inode 的記錄來指向；
  - 想要讀取該檔案，必須要經過目錄記錄的檔名來指向到正確的 inode 號碼才能讀取。
- 其實檔名只與目錄有關，但是檔案內容則與 inode 有關。那麼想一想， 有沒有可能有多個檔名對應到同一個 inode 號碼呢？有的！那就是 hard link 的由來。 所以簡單的說：hard link 只是在某個目錄下新增一筆檔名連結到某 inode 號碼的關連記錄而已。

實體連結與符號連結： ln

- Hard Link (實體連結, 硬式連結或實際連結)
- 假設我系統有個 /root/crontab 他是 /etc/crontab 的實體連結，也就是說這兩個檔名連結到同一個 inode ， 自然這兩個檔名的所有相關資訊都會一模一樣(除了檔名之外)。實際的情況可以如下所示：

```
[root@www ~]# ln /etc/crontab .   <==建立實體連結的指令
[root@www ~]# ll -i /etc/crontab /root/crontab
1912701 -rw-r--r-- 2 root root 255 Jan  6  2007 /etc/crontab
1912701 -rw-r--r-- 2 root root 255 Jan  6  2007 /root/crontab
```



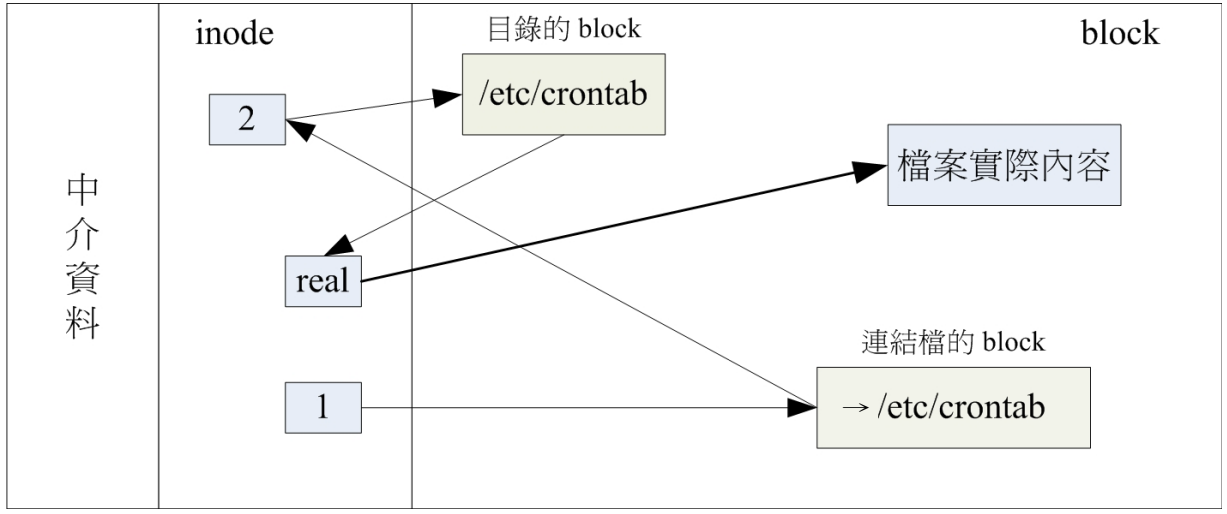
實體連結的檔案讀取示意圖

- 可以透過 1 或 2 的目錄之 inode 指定的 block 找到兩個不同的檔名，而不管使用哪個檔名均可以指到 real 那個 inode 去讀取到最終資料！那這樣有什麼好處呢？最大的好處就是『安全』！如同上圖中， 如果你將任何一個『檔名』刪除，其實 inode 與 block 都還是存在的！ 此時你可以透過另一個『檔名』來讀取到正確的檔案資料喔！此外，不論你使用哪個『檔名』來編輯， 最終的結果都會寫入到相同的 inode 與 block 中，因此均能進行資料的修改
- 事實上 hard link 應該僅能在單一檔案系統中進行的，應該是不能夠跨檔案系統才對！ 因為圖 2.2.1 就是在同一個 filesystem 上嘛！所以 hard link 是有限制的：
  - 不能跨 Filesystem
  - 不能 link 目錄

Symbolic Link (符號連結，亦即是捷徑)

- Symbolic link 就是在建立一個獨立的檔案，而這個檔案會讓資料的讀取指向他 link 的那個檔案的檔名！由於只是利用檔案來做為指向的動作， 所以，當來源檔被刪除之後，symbolic link 的檔案會『開不了』， 會一直說『無法開啟某檔案！』。實際上就是找不到原始『檔名』

```
[root@www ~]# ln -s /etc/crontab crontab2
[root@www ~]# ll -i /etc/crontab /root/crontab2
1912701 -rw-r--r-- 2 root root 255 Jan  6  2007 /etc/crontab
 654687 lrwxrwxrwx 1 root root  12 Oct 22 13:58 /root/crontab2 -> /etc/crontab
```



符號連結的檔案讀取示意圖

```
[root@www ~]# ln [-sf] 來源檔 目標檔
選項與參數：
-s   ：如果不加任何參數就進行連結，那就是hard link，至於 -s 就是symbolic link
-f   ：如果 目標檔 存在時，就主動的將目標檔直接移除後再建立！

範例一：將 /etc/passwd 複製到 /tmp 底下，並且觀察 inode 與 block
[root@www ~]# cd /tmp
[root@www tmp]# cp -a /etc/passwd .
[root@www tmp]# du -sb ; df -i .
18340    .   <==先注意一下這裡的容量是多少！
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hdc2            2560864  149738 2411126     6% /
# 利用 du 與 df 來檢查一下目前的參數~那個 du -sb
# 是計算整個 /tmp 底下有多少 bytes 的容量啦！

範例二：將 /tmp/passwd 製作 hard link 成為 passwd-hd 檔案，並觀察檔案與容量
[root@www tmp]# ln passwd passwd-hd
[root@www tmp]# du -sb ; df -i .
18340    .
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hdc2            2560864  149738 2411126     6% /
# 仔細看，即使多了一個檔案在 /tmp 底下，整個 inode 與 block 的容量並沒有改變！

[root@www tmp]# ls -il passwd*
586361 -rw-r--r-- 2 root root 1945 Sep 29 02:21 passwd
586361 -rw-r--r-- 2 root root 1945 Sep 29 02:21 passwd-hd
# 原來是指向同一個 inode 啊！這是個重點啊！另外，那個第二欄的連結數也會增加！
```

關於目錄的 link 數量

- 如果建立目錄時，他預設的 link 數量會是多少？ 讓我們來想一想，一個『空目錄』裡面至少會存在些什麼？就是存在 . 與 .. 這兩個目錄！ 那麼，當我們建立一個新目錄名稱為 /tmp/testing 時，基本上會有三個東西，那就是：
  - /tmp/testing
  - /tmp/testing/.
  - /tmp/testing/..

```
[root@www ~]# ls -ld /tmp
drwxrwxrwt 5 root root 4096 Oct 22 14:22 /tmp
[root@www ~]# mkdir /tmp/testing1
[root@www ~]# ls -ld /tmp
drwxrwxrwt 6 root root 4096 Oct 22 14:37 /tmp
[root@www ~]# ls -ld /tmp/testing1
drwxr-xr-x 2 root root 4096 Oct 22 14:37 /tmp/testing1
```

磁碟的分割、格式化、檢驗與掛載

- 對於一個系統管理者(root)而言，磁碟的的管理是相當重要的一環，尤其近來硬碟已經漸漸的被當成是消耗品了 ..... 如果我們想要在系統裡面新增一顆硬碟時，應該有哪些動作需要做的呢：
- 1.對磁碟進行分割，以建立可用的 partition ；
- 2.對該 partition 進行格式化(format)，以建立系統可用的 filesystem
- 3.若想要仔細一點，則可對剛剛建立好的 filesystem 進行檢驗；
- 4.在 Linux 系統上，需要建立掛載點 (亦即是目錄)，並將它掛載上來；

磁碟的分割、格式化、檢驗與掛載

```
[root@www ~]# fdisk [-l] 裝置名稱
選項與參數：
-l   ：輸出後面接的裝置所有的 partition 內容。若僅有 fdisk -l 時，
       則系統將會把整個系統內能夠搜尋到的裝置的 partition 均列出來。

範例：找出你系統中的根目錄所在磁碟，並查閱該硬碟內的相關資訊
[root@www ~]# df /                <==注意：重點在找出磁碟檔名而已
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hdc2              9920624   3823168   5585388  41% /

[root@www ~]# fdisk /dev/hdc  <==仔細看，不要加上數字喔！
The number of cylinders for this disk is set to 5005.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):    <==等待你的輸入！
```

```
Command (m for help): m   <== 輸入 m 後，就會看到底下這些指令介紹
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition              <==刪除一個partition
   l   list known partition types
   m   print this menu
   n   add a new partition             <==新增一個partition
   o   create a new empty DOS partition table
   p   print the partition table       <==在螢幕上顯示分割表
   q   quit without saving changes     <==不儲存離開fdisk程式
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit    <==將剛剛的動作寫入分割表
   x   extra functionality (experts only)
```

```
Command (m for help): p  <== 這裡可以輸出目前磁碟的狀態

Disk /dev/hdc: 41.1 GB, 41174138880 bytes          <==這個磁碟的檔名與容量
255 heads, 63 sectors/track, 5005 cylinders        <==磁頭、磁區與磁柱大小
Units = cylinders of 16065 * 512 = 8225280 bytes <==每個磁柱的大小

   Device Boot      Start         End      Blocks   Id  System
/dev/hdc1   *           1          13      104391   83  Linux
/dev/hdc2              14        1288    10241437+  83  Linux
/dev/hdc3            1289        1925     5116702+  83  Linux
/dev/hdc4            1926        5005    24740100    5  Extended
/dev/hdc5            1926        2052     1020096   82  Linux swap / Solaris
# 裝置檔名 開機區否 開始磁柱    結束磁柱  1K大小容量 磁碟分割槽內的系統

Command (m for help): q
# 想要不儲存離開嗎？按下 q 就對了！不要隨便按 w 啊！
```

磁碟分割： fdisk

- Device：裝置檔名，依據不同的磁碟介面/分割槽位置而變。
- Boot：是否為開機啟動區塊？通常 Windows 系統的 C 需要這塊！
- Start, End：這個分割槽在哪個磁柱號碼之間，可以決定此分割槽的大小；
- Blocks：就是以 1K 為單位的容量。如上所示，/dev/hdc1 大小為 104391K = 102MB
- ID, System：代表這個分割槽內的檔案系統應該是啥！不過這個項目只是一個提示而已，不見得真的代表此分割槽內的檔案系統喔！
- 從上表我們可以發現幾件事情：
- 整部磁碟還可以進行額外的分割，因為最大磁柱為 5005 ，但只使用到 2052 號而已；
- /dev/hdc5 是由 /dev/hdc4 分割出來的，因為 /dev/hdc4 為 Extended，且 /dev/hdc5 磁柱號碼在 /dev/hdc4 之內；

範例：查閱目前系統內的所有 partition 有哪些？

```
[root@www ~]# fdisk -l
Disk /dev/hdc: 41.1 GB, 41174138880 bytes
255 heads, 63 sectors/track, 5005 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdc1   *           1          13      104391   83  Linux
/dev/hdc2              14        1288    10241437+  83  Linux
/dev/hdc3            1289        1925     5116702+  83  Linux
/dev/hdc4            1926        5005    24740100    5  Extended
/dev/hdc5            1926        2052     1020096   82  Linux swap / Solaris

Disk /dev/sda: 8313 MB, 8313110528 bytes
59 heads, 58 sectors/track, 4744 cylinders
Units = cylinders of 3422 * 512 = 1752064 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1        4745     8118260    b  W95 FAT32
```

刪除磁碟分割槽

```
# 練習一： 先進入 fdisk 的畫面當中去！
[root@www ~]# fdisk /dev/hdc

# 練習二： 先看看整個分割表的情況是如何
Command (m for help): p

Disk /dev/hdc: 41.1 GB, 41174138880 bytes
255 heads, 63 sectors/track, 5005 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdc1   *           1          13      104391   83  Linux
/dev/hdc2              14        1288    10241437+  83  Linux
/dev/hdc3            1289        1925     5116702+  83  Linux
/dev/hdc4            1926        5005    24740100    5  Extended
/dev/hdc5            1926        2052     1020096   82  Linux swap / Solaris

# 練習三： 按下 d 給他刪除吧！
Command (m for help): d
Partition number (1-5): 4

Command (m for help): d
Partition number (1-4): 3
```

```
Command (m for help): p

Disk /dev/hdc: 41.1 GB, 41174138880 bytes
255 heads, 63 sectors/track, 5005 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdc1   *           1          13      104391   83  Linux
/dev/hdc2              14        1288    10241437+  83  Linux
# 因為 /dev/hdc5 是由 /dev/hdc4 所衍生出來的邏輯分割槽，因此 /dev/hdc4 被刪除，
# /dev/hdc5 就自動不見了！最終就會剩下兩個分割槽而已喔！

Command (m for help): q
# 鳥哥這裡僅是做一個練習而已，所以，按下 q 就能夠離開囉～
```

新增磁碟分割槽

```
# 練習一： 進入 fdisk 的分割軟體畫面中，並刪除所有分割槽：
[root@www ~]# fdisk /dev/hdc
Command (m for help): d
Partition number (1-5): 4

Command (m for help): d
Partition number (1-4): 3

Command (m for help): d
Partition number (1-4): 2

Command (m for help): d
Selected partition 1
# 由於最後僅剩下一個 partition ，因此系統主動選取這個 partition 刪除去！

# 練習二： 開始新增，我們先新增一個 Primary  的分割槽，且指定為 4 號看看！
Command (m for help): n
Command action                  <==因為是全新磁碟，因此只會問extended/primary而已
   e   extended
   p   primary partition (1-4)
p                               <==選擇 Primary 分割槽
Partition number (1-4): 4 <==設定為 4 號！
First cylinder (1-5005, default 1): <==直接按下[enter]按鍵決定！
Using default value 1                  <==啟始磁柱就選用預設值！
Last cylinder or +size or +sizeM or +sizeK (1-5005, default 5005): +512M
```

```
# 練習三： 繼續新增一個，這次我們新增 Extended 的分割槽好了！
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e     <==選擇的是 Extended 喔！
Partition number (1-4): 1
First cylinder (64-5005, default 64): <=[enter]
Using default value 64
Last cylinder or +size or +sizeM or +sizeK (64-5005, default 5005): <=[enter]
Using default value 5005
# 還記得我們在第三章的磁碟分割表曾經談到過的，延伸分割最好能夠包含所有
# 未分割的區間；所以在這個練習中，我們將所有未配置的磁柱都給了這個分割槽喔！
# 所以在開始/結束磁柱的位置上，按下兩個[enter]用預設值即可！

Command (m for help): p

Disk /dev/hdc: 41.1 GB, 41174138880 bytes
255 heads, 63 sectors/track, 5005 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdc1              64        5005    39696615    5  Extended
/dev/hdc4               1          63      506016   83  Linux
# 如上所示，所有的磁柱都在 /dev/hdc1 裡面囉！
```

```
# 練習四： 這次我們隨便新增一個 2GB 的分割槽看看！
Command (m for help): n
Command action
   l   logical (5 or over)      <==因為已有 extended ，所以出現 logical 分割槽
   p   primary partition (1-4)
p    <==偷偷玩一下，能否新增主要分割槽
Partition number (1-4): 2
No free sectors available    <==肯定不行！因為沒有多餘的磁柱可供配置

Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l    <==乖乖使用邏輯分割槽吧！
First cylinder (64-5005, default 64): <=[enter]
Using default value 64
Last cylinder or +size or +sizeM or +sizeK (64-5005, default 5005): +2048M

Command (m for help): p

Disk /dev/hdc: 41.1 GB, 41174138880 bytes
255 heads, 63 sectors/track, 5005 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdc1              64        5005    39696615    5  Extended
/dev/hdc4               1          63      506016   83  Linux
/dev/hdc5              64         313     2008093+  83  Linux
# 這樣就新增了 2GB 的分割槽，且由於是 logical ，所以由 5 號開始！
Command (m for help): q
# 鳥哥這裡僅是做一個練習而已，所以，按下 q 就能夠離開囉～
```

磁碟格式化

mkfs

[root@www ~]# **mkfs [-t** 檔案系統格式**]** 裝置檔名
選項與參數：
-t ：可以接檔案系統格式，例如 ext3, ext2, vfat 等(系統有支援才會生效)

```
[root@www ~]# mkfs [-t 檔案系統格式] 裝置檔名
選項與參數：
-t   ：可以接檔案系統格式，例如 ext3, ext2, vfat 等(系統有支援才會生效)

範例一：請將上個小節當中所製作出來的 /dev/hdc6 格式化為 ext3 檔案系統
[root@www ~]# mkfs -t ext3 /dev/hdc6
mke2fs 1.39 (29-May-2006)
Filesystem label=                    <==這裡指的是分割槽的名稱(label)
OS type: Linux
Block size=4096 (log=2)              <==block 的大小設定為 4K
Fragment size=4096 (log=2)
251392 inodes, 502023 blocks         <==由此設定決定的inode/block數量
25101 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=515899392
16 block groups
32768 blocks per group, 32768 fragments per group
15712 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done <==有日誌記錄
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
# 這樣就建立起來我們所需要的 Ext3 檔案系統了！簡單明瞭！

[root@www ~]# mkfs[tab][tab]
mkfs          mkfs.cramfs  mkfs.ext2    mkfs.ext3    mkfs.msdos   mkfs.vfat
# 按下兩個[tab]，會發現 mkfs 支援的檔案格式如上所示！可以格式化 vfat 喔！
```

mke2fs

[root@www ~]# **mke2fs [-b block大小] [-i block大小] [-L 標頭] [-cj] 裝置**
選項與參數：
-b ：可以設定每個 block 的大小，目前支援 1024, 2048, 4096 bytes 三種；
-i ：多少容量給予一個 inode 呢？
-c ：檢查磁碟錯誤，僅下達一次 -c 時，會進行快速讀取測試；
　　如果下達兩次 -c -c 的話，會測試讀寫(read-write)，會很慢～
-L ：後面可以接標頭名稱 (Label)，這個 label 是有用的喔！e2label指令介紹會談到～
-j ：本來 mke2fs 是 EXT2 ，加上 -j 後，會主動加入 journal 而成為 EXT3。

- mke2fs 是一個很詳細但是很麻煩的指令！因為裡面的細部設定太多了！現在我們進行如下的假設：
  - 這個檔案系統的標頭設定為：vbird_logical
  - 我的 block 指定為 2048 大小；
  - 每 8192 bytes 分配一個 inode ；
  - 建置為 journal 的 Ext3 檔案系統。

開始格式化 /dev/hdc6 結果會變成如下所示：

```
[root@www ~]# mke2fs -j -L "vbird_logical" -b 2048 -i 8192 /dev/hdc6
mke2fs 1.39 (29-May-2006)
Filesystem label=vbird_logical
OS type: Linux
Block size=2048 (log=1)
Fragment size=2048 (log=1)
251968 inodes, 1004046 blocks
50202 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=537919488
62 block groups
16384 blocks per group, 16384 fragments per group
4064 inodes per group
Superblock backups stored on blocks:
        16384, 49152, 81920, 114688, 147456, 409600, 442368, 802816
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
# 比較看看，跟上面的範例用預設值的結果，有什麼不一樣的啊？
```

磁碟檢驗： fsck, badblocks

- 由於系統在運作時誰也說不準啥時硬體或者是電源會有問題，所以『當機』可能是難免的情況(不管是硬體還是軟體)。 現在我們知道檔案系統運作時會有硬碟與記憶體資料非同步的狀況發生，因此莫名其妙的當機非常可能導致檔案系統的錯亂。 問題來啦，如果檔案系統真的發生錯亂的話，那該如何是好？

```
[root@www ~]# fsck [-t 檔案系統] [-ACay] 裝置名稱
選項與參數：
-t   ：如同 mkfs 一樣，fsck 也是個綜合軟體而已！因此我們同樣需要指定檔案系統。
       不過由於現今的 Linux 太聰明了，他會自動的透過 superblock 去分辨檔案系統，
       因此通常可以不需要這個選項的囉！請看後續的範例說明。
-A   ：依據 /etc/fstab 的內容，將需要的裝置掃瞄一次。/etc/fstab 於下一小節說明，
       通常開機過程中就會執行此一指令了。
-a   ：自動修復檢查到的有問題的磁區，所以你不用一直按 y 囉！
-y   ：與 -a 類似，但是某些 filesystem 僅支援 -y 這個參數！
-C   ：可以在檢驗的過程當中，使用一個長條圖來顯示目前的進度！

EXT2/EXT3 的額外選項功能：(e2fsck 這支指令所提供)
-f   ：強制檢查！一般來說，如果 fsck 沒有發現任何 unclean 的旗標，不會主動進入
       細部檢查的，如果您想要強制 fsck 進入細部檢查，就得加上 -f 旗標囉！
-D   ：針對檔案系統下的目錄進行最佳化配置。
```

磁碟檢驗： fsck

```
範例一：強制的將前面我們建立的 /dev/hdc6 這個裝置給他檢驗一下！
[root@www ~]# fsck -C -f -t ext3 /dev/hdc6
fsck 1.39 (29-May-2006)
e2fsck 1.39 (29-May-2006)
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
vbird_logical: 11/251968 files (9.1% non-contiguous), 36926/1004046 blocks
# 如果沒有加上 -f 的選項，則由於這個檔案系統不曾出現問題，
# 檢查的經過非常快速！若加上 -f 強制檢查，才會一項一項的顯示過程。

範例二：系統有多少檔案系統支援的 fsck 軟體？
[root@www ~]# fsck[tab][tab]
fsck          fsck.cramfs   fsck.ext2     fsck.ext3     fsck.msdos    fsck.vfat
```

badblocks

```
[root@www ~]# badblocks -[svw] 裝置名稱
選項與參數：
-s   ：在螢幕上列出進度
-v   ：可以在螢幕上看到進度
-w   ：使用寫入的方式來測試，建議不要使用此一參數，尤其是待檢查的裝置已有檔案時！
[root@www ~]# badblocks -sv /dev/hdc6
Checking blocks 0 to 2008093
Checking for bad blocks (read-only test): done
Pass completed, 0 bad blocks found.
```

- 掛載點的意義當中提過掛載點是目錄， 而這個目錄是進入磁碟分割槽(其實是檔案系統啦！)的入口就是了。不過要進行掛載前，你最好先確定幾件事：
  - 單一檔案系統不應該被重複掛載在不同的掛載點(目錄)中；
  - 單一目錄不應該重複掛載多個檔案系統；
  - 要作為掛載點的目錄，理論上應該都是空目錄才是。
- 掛載Ext2/Ext3檔案系統
  - /etc/filesystems：系統指定的測試掛載檔案系統類型；
  - /proc/filesystems：Linux系統已經載入的檔案系統類型。

```
[root@www ~]# mount -a
[root@www ~]# mount [-l]
[root@www ~]# mount [-t 檔案系統] [-L Label名] [-o 額外選項] \
 [-n]  裝置檔名  掛載點
選項與參數：
-a   ：依照設定檔 /etc/fstab 的資料將所有未掛載的磁碟都掛載上來
-l   ：單純的輸入 mount 會顯示目前掛載的資訊。加上 -l 可增列 Label 名稱！
-t   ：與 mkfs 的選項非常類似的，可以加上檔案系統種類來指定欲掛載的類型。
       常見的 Linux 支援類型有：ext2, ext3, vfat, reiserfs, iso9660(光碟格式),
       nfs, cifs, smbfs(此三種為網路檔案系統類型)
-n   ：在預設的情況下，系統會將實際掛載的情況即時寫入 /etc/mtab 中，以利其他程式
       的運作。但在某些情況下(例如單人維護模式)為了避免問題，會刻意不寫入。
       此時就得要使用這個 -n 的選項了。
-L   ：系統除了利用裝置檔名 (例如 /dev/hdc6) 之外，還可以利用檔案系統的標頭名稱
       (Label)來進行掛載。最好為你的檔案系統取一個獨一無二的名稱吧！
-o   ：後面可以接一些掛載時額外加上的參數！比方說帳號、密碼、讀寫權限等：
       ro, rw:          掛載檔案系統成為唯讀(ro) 或可讀寫(rw)
       async, sync:     此檔案系統是否使用同步寫入 (sync) 或非同步 (async) 的
                        記憶體機制，請參考檔案系統運作方式。預設為 async。
       auto, noauto:    允許此 partition 被以 mount -a 自動掛載(auto)
       dev, nodev:      是否允許此 partition 上，可建立裝置檔案？ dev 為可允許
       suid, nosuid:    是否允許此 partition 含有 suid/sgid 的檔案格式？
       exec, noexec:    是否允許此 partition 上擁有可執行 binary 檔案？
       user, nouser:    是否允許此 partition 讓任何使用者執行 mount ？一般來說，
                        mount 僅有 root 可以進行，但下達 user 參數，則可讓
                        一般 user 也能夠對此 partition 進行 mount 。
       defaults:        預設值為：rw, suid, dev, exec, auto, nouser, and async
       remount:         重新掛載，這在系統出錯，或重新更新參數時，很有用！
```

掛載EXT2/EXT3檔案系統

```
範例一：用預設的方式，將剛剛建立的 /dev/hdc6 掛載到 /mnt/hdc6 上面！
[root@www ~]# mkdir /mnt/hdc6
[root@www ~]# mount /dev/hdc6 /mnt/hdc6
[root@www ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
.....中間省略.....
/dev/hdc6              1976312     42072   1833836   3% /mnt/hdc6
# 看起來，真的有掛載！且檔案大小約為 2GB 左右啦！
```

- 由於檔案系統幾乎都有 superblock ， 我們的 Linux 可以透過分析 superblock 搭配 Linux 自己的驅動程式去測試掛載， 如果成功的套和了，就立刻自動的使用該類型的檔案系統掛載起來啊！ 那麼系統有沒有指定哪些類型的 filesystem 才需要進行上述的掛載測試呢？ 主要是參考底下這兩個檔案：
  - /etc/filesystems：系統指定的測試掛載檔案系統類型；
  - /proc/filesystems：Linux系統已經載入的檔案系統類型

```
範例二：觀察目前『已掛載』的檔案系統，包含各檔案系統的Label名稱
[root@www ~]# mount -l
/dev/hdc2 on / type ext3 (rw) [/1]
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/hdc3 on /home type ext3 (rw) [/home]
/dev/hdc1 on /boot type ext3 (rw) [/boot]
tmpfs on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/hdc6 on /mnt/hdc6 type ext3 (rw) [vbird_logical]
# 除了實際的檔案系統外，很多特殊的檔案系統(proc/sysfs...)也會被顯示出來！
# 值得注意的是，加上 -l 選項可以列出如上特殊字體的標頭(label)喔
```

掛載 CD 或 DVD 光碟

範例三：將你用來安裝 Linux 的 CentOS 原版光碟拿出來掛載！

```
[root@www ~]# mkdir /media/cdrom
[root@www ~]# mount -t iso9660 /dev/cdrom /media/cdrom
[root@www ~]# mount /dev/cdrom /media/cdrom
# 你可以指定 -t iso9660 這個光碟片的格式來掛載，也可以讓系統自己去測試掛載！
# 所以上述的指令只要做一個就夠了！但是目錄的建立初次掛載時必須要進行喔！

[root@www ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
.....中間省略.....
/dev/hdd              4493152   4493152         0 100% /media/cdrom
# 因為我的光碟機使用的是 /dev/hdd 的 IDE 介面之故！
```

格式化與掛載軟碟

- 軟碟的格式化可以直接使用 mkfs 即可。但是軟碟也是可以格式化成為 ext3 或 vfat 格式的。

```
範例四：格式化後掛載軟碟到 /media/floppy/ 目錄中。
[root@www ~]# mkfs -t vfat /dev/fd0
# 我們格式化軟碟成為 Windows/Linux 可共同使用的 FAT 格式吧！
[root@www ~]# mkdir /media/floppy
[root@www ~]# mount -t vfat /dev/fd0 /media/floppy
[root@www ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
.....中間省略.....
/dev/fd0                  1424       164      1260  12% /media/floppy
```

掛載隨身碟

```
範例五：找出你的隨身碟裝置檔名，並掛載到 /mnt/flash 目錄中
[root@www ~]# fdisk -l
.....中間省略.....
Disk /dev/sda: 8313 MB, 8313110528 bytes
59 heads, 58 sectors/track, 4744 cylinders
Units = cylinders of 3422 * 512 = 1752064 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1        4745     8118260    b  W95 FAT32
# 從上的特殊字體，可得知磁碟的大小以及裝置檔名，知道是 /dev/sda1

[root@www ~]# mkdir /mnt/flash
[root@www ~]# mount -t vfat -o iocharset=cp950 /dev/sda1 /mnt/flash
[root@www ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
.....中間省略.....
/dev/sda1              8102416   4986228   3116188  62% /mnt/flash
```

重新掛載根目錄與掛載不特定目錄

- 整個目錄樹最重要的地方就是根目錄了，所以根目錄根本就不能夠被卸載的！問題是，如果你的掛載參數要改變， 或者是根目錄出現『唯讀』狀態時，如何重新掛載呢？最可能的處理方式就是重新開機 (reboot)！ 不過你也可以這樣做：

```
範例六：將 / 重新掛載，並加入參數為 rw 與 auto
[root@www ~]# mount -o remount,rw,auto /
```

- 我們也可以利用 mount 來將某個目錄掛載到另外一個目錄去喔！這並不是掛載檔案系統，而是額外掛載某個目錄的方法！

```
範例七：將 /home 這個目錄暫時掛載到 /mnt/home 底下：
[root@www ~]# mkdir /mnt/home
[root@www ~]# mount --bind /home /mnt/home
[root@www ~]# ls -lid /home/ /mnt/home
2 drwxr-xr-x 6 root root 4096 Sep 29 02:21 /home/
2 drwxr-xr-x 6 root root 4096 Sep 29 02:21 /mnt/home
[root@www ~]# mount -l
/home on /mnt/home type none (rw,bind)
```

umount (將裝置檔案卸載)

```
[root@www ~]# umount [-fn] 裝置檔名或掛載點
選項與參數：
-f ：強制卸載！可用在類似網路檔案系統 (NFS) 無法讀取到的情況下；
-n ：不更新 /etc/mtab 情況下卸載。
```

```
範例八：將本章之前自行掛載的檔案系統全部卸載：
[root@www ~]# mount
.....前面省略.....
/dev/hdc6 on /mnt/hdc6 type ext3 (rw)
/dev/hdd on /media/cdrom type iso9660 (rw)
/dev/sda1 on /mnt/flash type vfat (rw,iocharset=cp950)
/home on /mnt/home type none (rw,bind)
# 先找一下已經掛載的檔案系統，如上所示，特殊字體即為剛剛掛載的裝置囉！
[root@www ~]# umount /dev/hdc6       <==用裝置檔名來卸載
[root@www ~]# umount /media/cdrom    <==用掛載點來卸載
[root@www ~]# umount /mnt/flash      <==因為掛載點比較好記憶！
[root@www ~]# umount /dev/fd0        <==用裝置檔名較好記！
[root@www ~]# umount /mnt/home       <==一定要用掛載點！因為掛載的是目錄
```

- 如果你遇到這樣的情況：

```
[root@www ~]# mount /dev/cdrom /media/cdrom
[root@www ~]# cd /media/cdrom
[root@www cdrom]# umount /media/cdrom
umount: /media/cdrom: device is busy
umount: /media/cdrom: device is busy
```

- 由於你目前正在 /media/cdrom/ 的目錄內，也就是說其實『你正在使用該檔案系統』的意思！ 所以自然無法卸載這個裝置！那該如何是好？就『離開該檔案系統的掛載點』即可。以上述的案例來說， 你可以使用『 cd / 』回到根目錄，就能夠卸載 /media/cdrom

使用 Label Name 進行掛載的方法

```
範例九：找出 /dev/hdc6 的 label name，並用 label 掛載到 /mnt/hdc6
[root@www ~]# dumpe2fs -h /dev/hdc6
Filesystem volume name:   vbird_logical
.....底下省略.....
# 找到啦！標頭名稱為 vbird_logical 囉！
[root@www ~]# mount -L "vbird_logical" /mnt/hdc6
```

mknod

- 在 Linux 底下所有的裝置都以檔案來代表吧！但是那個檔案如何代表該裝置呢？就是透過檔案的 major 與 minor 數值來替代的～所以，那個 major 與 minor 數值是有特殊意義的，不是隨意設定的

磁碟檔名	Major	Minor
/dev/hda	3	0~63
/dev/hdb	3	64~127
/dev/sda	8	0-15
/dev/sdb	8	16-31

e2label

- 那如果格式化完畢後想要修改標頭呢？就用這個 e2label 來修改了。目前 CentOS 的設定檔，也就是那個 /etc/fstab 檔案的設定都預設使用 Label name

```
[root@www ~]# e2label 裝置名稱  新的Label名稱
範例一：將 /dev/hdc6 的標頭改成 my_test 並觀察是否修改成功？
[root@www ~]# dumpe2fs -h /dev/hdc6
Filesystem volume name:   vbird_logical  <==原本的標頭名稱
.....底下省略.....
[root@www ~]# e2label /dev/hdc6 "my_test"
[root@www ~]# dumpe2fs -h /dev/hdc6
Filesystem volume name:   my_test        <==改過來啦！
.....底下省略.....
```

hdparm

- 如果你的硬碟是 IDE 介面的，那麼這個指令可以幫助你設定一些進階參數

[root@www ~]# **hdparm [-icdmXTt] 裝置名稱**
選項與參數：
-i ：將核心偵測到的硬碟參數顯示出來！
-c ：設定 32-bit (32位元)存取模式。這個 32 位元存取模式指的是在硬碟在與 PCI 介面之間傳輸的模式，而硬碟本身是依舊以 16 位元模式在跑的！ 預設的情況下，這個設定值都會被打開，建議直接使用 c1 即可！
-d ：設定是否啟用 dma 模式，-d1 為啟動，-d0 為取消；
-m ：設定同步讀取多個 sector 的模式。一般來說，設定此模式，可降低系統因為讀取磁碟而損耗的效能～不過，WD 的硬碟則不怎麼建議設定此值～ 一般來說，設定為 16/32 是最佳化，不過，WD 硬碟建議值則是 4/8 。 這個值的最大值，可以利用 hdparm -i /dev/hda 輸出的 MaxMultSect 來設定喔！一般如果不曉得，設定 16 是合理的！
-X ：設定 UtraDMA 的模式，一般來說，UDMA 的模式值加 64 即為設定值。 並且，硬碟與主機板晶片必須要同步，所以，取最小的那個。一般來說：
33 MHz DMA mode 0~2 (X64~X66)
66 MHz DMA mode 3~4 (X67~X68)
100MHz DMA mode 5 (X69)
如果你的硬碟上面顯示的是 UATA 100 以上的，那麼設定 X69 也不錯！
-T ：測試暫存區 cache 的存取效能
-t ：測試硬碟的實際存取效能 (較正確！)

設定開機掛載

- 開機掛載 /etc/fstab 及 /etc/mtab
  - 根目錄 / 是必須掛載的，而且一定要先於其它 mount point 被掛載進來。
  - 其它 mount point 必須為已建立的目錄，可任意指定，但一定要遵守必須的系統目錄架構原則
  - 所有 mount point 在同一時間之內，只能掛載一次。
  - 所有 partition 在同一時間之內，只能掛載一次。
  - 如若進行卸載，你必須先將工作目錄移到 mount point(及其子目錄) 之外。

設定開機掛載

- 第一欄：磁碟裝置檔名或該裝置的 Label
- 第二欄：掛載點 (mount point)
- 第三欄：磁碟分割槽的檔案系統
- 第四欄：檔案系統參數
- 第五欄：能否被 dump 備份指令作用

```
[root@www ~]# cat /etc/fstab
# Device        Mount point   filesystem parameters    dump fsck
LABEL=/1          /           ext3       defaults        1 1
LABEL=/home       /home       ext3       defaults        1 2
LABEL=/boot       /boot       ext3       defaults        1 2
tmpfs               /dev/shm    tmpfs       defaults          0 0
devpts              /dev/pts    devpts      gid=5,mode=620    0 0
sysfs               /sys        sysfs       defaults          0 0
proc                /proc       proc        defaults          0 0
LABEL=SWAP-hdc5     swap        swap        defaults          0 0
# 上述特殊字體的部分與實際磁碟有關！其他則是虛擬檔案系統或
# 與記憶體置換空間 (swap) 有關。
```

第四欄—參數設定

參數	內容意義
async/sync 非同步/同步	設定磁碟是否以非同步方式運作！預設為 async(效能較佳)
auto/noauto 自動/非自動	當下達 mount -a 時，此檔案系統是否會被主動測試掛載。預設為 auto。
rw/ro 可讀寫/唯讀	讓該分割槽以可讀寫或者是唯讀的型態掛載上來，如果你想要分享的資料是不給使用者隨意變更的， 這裡也能夠設定為唯讀。則不論在此檔案系統的檔案是否設定 w 權限，都無法寫入喔！
exec/noexec 可執行/不可執行	限制在此檔案系統內是否可以進行『執行』的工作？如果是純粹用來儲存資料的， 那麼可以設定為 noexec 會比較安全，相對的，會比較麻煩！
user/nouser 允許/不允許使用者掛載	是否允許使用者使用 mount 指令來掛載呢？一般而言，我們當然不希望一般身份的 user 能使用 mount 囉，因為太不安全了，因此這裡應該要設定為 nouser 囉！
suid/nosuid 具有/不具有 suid 權限	該檔案系統是否允許 SUID 的存在？如果不是執行檔放置目錄，也可以設定為 nosuid 來取消這個功能！
usrquota	注意名稱是『 usrquota 』不要拼錯了！這個是在啟動 filesystem 支援磁碟配額模式，更多資料我們在第四篇再談。
grpquota	注意名稱是『grpquota』，啟動 filesystem 對群組磁碟配額模式的支援。
defaults	同時具有 rw, suid, dev, exec, auto, nouser, async 等參數。基本上，預設情況使用 defaults 設定即可！

- 假設我們要將 /dev/hdc6 每次開機都自動掛載到 /mnt/hdc6 ，該如何進行？

```
[root@www ~]# nano /etc/fstab
/dev/hdc6  /mnt/hdc6    ext3    defaults   1 2
```

```
[root@www ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hdc6              1976312     42072   1833836   3% /mnt/hdc6
# 竟然不知道何時被掛載了？趕緊給他卸載先！

[root@www ~]# umount /dev/hdc6
```

```
[root@www ~]# mount -a
[root@www ~]# df
```

掛載光碟/DVD映象檔

```
[root@www ~]# ll -h /root/centos5.2_x86_64.iso
-rw-r--r-- 1 root root 4.3G Oct 27 17:34 /root/centos5.2_x86_64.iso
# 看到上面的結果吧！這個檔案就是映象檔，檔案非常的大吧！

[root@www ~]# mkdir /mnt/centos_dvd
[root@www ~]# mount -o loop /root/centos5.2_x86_64.iso /mnt/centos_dvd
[root@www ~]# df
Filesystem            1K-blocks      Used Available Use% Mounted on
/root/centos5.2_x86_64.iso
                        4493152   4493152         0 100% /mnt/centos_dvd
# 就是這個項目！ .iso 映象檔內的所有資料可以在 /mnt/centos_dvd 看到！

[root@www ~]# ll /mnt/centos_dvd
total 584
drwxr-xr-x 2 root root 522240 Jun 24 00:57 CentOS <==瞧！就是DVD的內容啊！
-rw-r--r-- 8 root root    212 Nov 21  2007 EULA
-rw-r--r-- 8 root root  18009 Nov 21  2007 GPL
drwxr-xr-x 4 root root   2048 Jun 24 00:57 images
.....底下省略.....

[root@www ~]# umount /mnt/centos_dvd/
# 測試完成！記得將資料給他卸載！
```

建立大檔案以製作 LOOP 裝置檔案

```
[root@www ~]# dd if=/dev/zero of=/home/loopdev bs=1M count=512
512+0 records in    <==讀入 512 筆資料
512+0 records out   <==輸出 512 筆資料
536870912 bytes (537 MB) copied, 12.3484 seconds, 43.5 MB/s
# 這個指令的簡單意義如下：
# if 是 input file ，輸入檔案。那個 /dev/zero 是會一直輸出 0 的裝置！
# of 是 output file ，將一堆零寫入到後面接的檔案中。
# bs 是每個 block 大小，就像檔案系統那樣的 block 意義；
# count 則是總共幾個 bs 的意思。

[root@www ~]# ll -h /home/loopdev
-rw-r--r-- 1 root root 512M Oct 28 02:29 /home/loopdev
```

dd 就好像在疊磚塊一樣，將 512 塊，每塊 1MB 的磚塊堆疊成為一個大檔案 (/home/loopdev) ！ 最終就會出現一個 512MB 的檔案！

格式化

```
[root@www ~]# mkfs -t ext3 /home/loopdev
mke2fs 1.39 (29-May-2006)
/home/loopdev is not a block special device.
Proceed anyway? (y,n) y  <==由於不是正常的裝置，所以這裡會提示你！
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
131072 inodes, 524288 blocks
26214 blocks (5.00%) reserved for the super user
.....以下省略.....
```

掛載

```
[root@www ~]# mount -o loop /home/loopdev /media/cdrom/
[root@www ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/home/loopdev           507748     18768    462766   4% /media/cdrom
```

- 透過這個簡單的方法，感覺上你就可以在原本的分割槽在不更動原有的環境下製作出你想要的分割槽就是了！這東西很好用的！尤其是想要玩 Linux 上面的『虛擬機器』的話， 也就是以一部 Linux 主機再切割成為數個獨立的主機系統時，類似VMware 這類的軟體， 在 Linux 上使用 xen 這個軟體，他就可以配合這種 loop device 的檔案類型來進行根目錄的掛載

記憶體置換空間(swap)之建置

- 現在想像一個情況，你已經將系統建立起來了，此時卻才發現你沒有建置 swap ~ 那該如何是好呢？
  - 設定一個 swap partition
  - 建立一個虛擬記憶體的檔案

使用實體分割槽建置swap

1. 分割：先使用 fdisk 在你的磁碟中分割中一個分割槽給系統作為 swap 。由於 Linux 的 fdisk 預設會將分割槽的 ID 設定為 Linux 的檔案系統，所以你可能還得要設定一下 system ID 就是了。
2. 格式化：利用建立 swap 格式的『mkswap 裝置檔名』就能夠格式化該分割槽成為 swap 格式囉！
3. 使用：最後將該 swap 裝置啟動，方法為：『swapon 裝置檔名』。
4. 觀察：最終透過 free 這個指令來觀察一下記憶體的用量吧！

```
[root@www ~]# fdisk /dev/hdc
Command (m for help): n
First cylinder (2303-5005, default 2303):  <==這裡按[enter]
Using default value 2303
Last cylinder or +size or +sizeM or +sizeK (2303-5005, default 5005): +256M

Command (m for help): p

   Device Boot      Start         End      Blocks   Id  System
.....中間省略.....
/dev/hdc6            2053        2302     2008093+  83  Linux
/dev/hdc7            2303        2334      257008+  83  Linux <==新增的項目

Command (m for help): t                  <==修改系統 ID
Partition number (1-7): 7                <==從上結果看到的，七號partition
Hex code (type L to list codes): 82 <==改成 swap 的 ID
Changed system type of partition 7 to 82 (Linux swap / Solaris)

Command (m for help): p

   Device Boot      Start         End      Blocks   Id  System
.....中間省略.....
/dev/hdc6            2053        2302     2008093+  83  Linux
/dev/hdc7            2303        2334      257008+  82  Linux swap / Solaris

Command (m for help): w
# 此時就將 partition table 更新了！

[root@www ~]# partprobe
# 這個玩意兒很重要的啦！不要忘記讓核心更新 partition table 喔！
```

開始建置 swap 格式

```
[root@www ~]# mkswap /dev/hdc7
Setting up swapspace version 1, size = 263172 kB  <==非常快速！
```

開始觀察與載入

```
[root@www ~]# free
             total       used       free     shared    buffers     cached
Mem:        742664     684592      58072          0      43820     497144
-/+ buffers/cache:     143628     599036
Swap:      1020088         96    1019992
# 我有 742664K 的實體記憶體，使用 684592K 剩餘 58072K ，使用掉的記憶體有
# 43820K / 497144K 用在緩衝/快取的用途中。
# 至於 swap 已經存在了 1020088K 囉！這樣會看了吧？！

[root@www ~]# swapon /dev/hdc7
[root@www ~]# free
             total       used       free     shared    buffers     cached
Mem:        742664     684712      57952          0      43872     497180
-/+ buffers/cache:     143660     599004
Swap:      1277088         96    1276992  <==有增加囉！看到否？

[root@www ~]# swapon -s
Filename                        Type            Size    Used    Priority
/dev/hdc5                       partition       1020088 96      -1
/dev/hdc7                       partition       257000  0       -2
# 上面列出目前使用的 swap 裝置有哪些的意思！
```

使用檔案建置swap

1. 使用 dd 這個指令來新增一個 128MB 的檔案在 /tmp 底下
2. 使用 mkswap 將 /tmp/swap 這個檔案格式化為 swap 的檔案格式
3. 使用 swapon 來將 /tmp/swap 啟動囉！
4. 使用 swapoff 關掉 swap file

```
[root@www ~]# dd if=/dev/zero of=/tmp/swap bs=1M count=128
128+0 records in
128+0 records out
134217728 bytes (134 MB) copied, 1.7066 seconds, 78.6 MB/s

[root@www ~]# ll -h /tmp/swap
-rw-r--r-- 1 root root 128M Oct 28 15:33 /tmp/swap


[root@www ~]# mkswap /tmp/swap
Setting up swapspace version 1, size = 134213 kB
# 這個指令下達時請『特別小心』，因為下錯字元控制，將可能使您的檔案系統掛掉！

[root@www ~]# free
             total       used       free     shared    buffers     cached
Mem:        742664     450860     291804          0      45584     261284
-/+ buffers/cache:     143992     598672
Swap:      1277088         96    1276992

[root@www ~]# swapon /tmp/swap
[root@www ~]# free
             total       used       free     shared    buffers     cached
Mem:        742664     450860     291804          0      45604     261284
-/+ buffers/cache:     143972     598692
Swap:      1408152         96    1408056

[root@www ~]# swapon -s
Filename                        Type            Size    Used    Priority
/dev/hdc5                       partition       1020088 96      -1
/dev/hdc7                       partition       257000  0       -2
/tmp/swap                       file            131064  0       -3
```

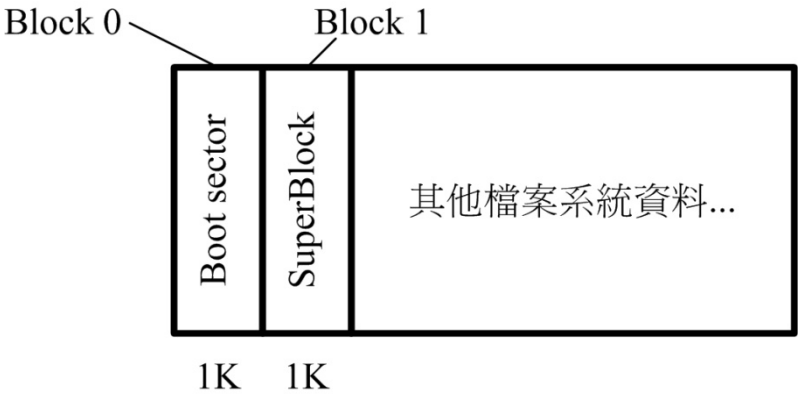
```
[root@www ~]# swapoff /tmp/swap
[root@www ~]# swapoff /dev/hdc7
[root@www ~]# free
             total       used       free     shared    buffers     cached
Mem:        742664     450860     291804          0      45660     261284
-/+ buffers/cache:     143916     598748
Swap:      1020088         96    1019992  <==回復成最原始的樣子了！
```

swap使用上的限制

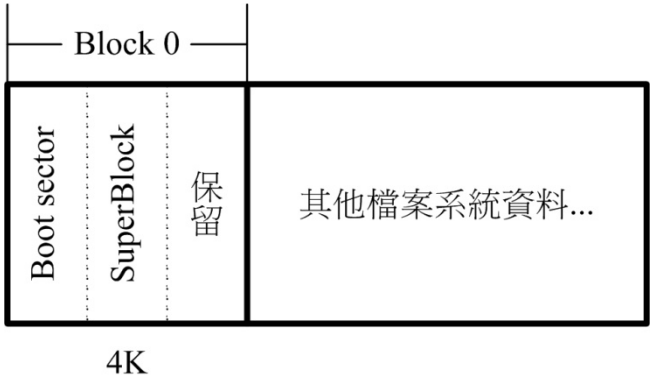
- 在核心 2.4.10 版本以後，單一 swap 量已經沒有 2GB 的限制了，
- 但是，最多還是僅能建立到 32 個 swap 的數量！
- 而且，由於目前 x86_64 (64位元) 最大記憶體定址到 64GB，因此，swap 總量最大也是僅能達 64GB 就是了！

檔案系統的特殊觀察與操作

- boot sector 與 superblock 的關係
  - superblock 的大小為 1024 bytes；
  - superblock 前面需要保留 1024 bytes 下來，以讓開機管理程式可以安裝。



1K block 的 boot sector 示意圖



4K block 的 boot sector 示意圖

磁碟空間之浪費問題

```
[root@www ~]# ll -s
total 104
 8 -rw------- 1 root root  1474 Sep  4 18:27 anaconda-ks.cfg
 8 -rw-r--r-- 2 root root   255 Jan  6  2007 crontab
 4 lrwxrwxrwx 1 root root    12 Oct 22 13:58 crontab2 -> /etc/crontab
48 -rw-r--r-- 1 root root 42304 Sep  4 18:26 install.log
12 -rw-r--r-- 1 root root  5661 Sep  4 18:25 install.log.syslog
 4 -rw-r--r-- 1 root root     0 Sep 27 00:25 test1
 8 drwxr-xr-x 2 root root  4096 Sep 27 00:25 test2
 4 -rw-rw-r-- 1 root root     0 Sep 27 00:36 test3
 8 drwxrwxr-x 2 root root  4096 Sep 27 00:36 test4
```

```
[root@www ~]# du -sb /etc
108360494       /etc    <==單位是 bytes 喔！
[root@www ~]# du -sm /etc
118     /etc            <==單位是 Mbytes 喔！
```

利用 GNU 的 parted 進行分割行為

```
[root@www ~]# parted [裝置] [指令 [參數]]
選項與參數：
指令功能：
新增分割：mkpart [primary|logical|extended] [ext3|vfat] 開始 結束
分割表   ：print
刪除分割：rm [partition]
範例一：以 parted 列出目前本機的分割表資料
[root@www ~]# parted /dev/hdc print
Model: IC35L040AVER07-0 (ide)                  <==硬碟介面與型號
Disk /dev/hdc: 41.2GB                          <==磁碟檔名與容量
Sector size (logical/physical): 512B/512B      <==每個磁區的大小
Partition Table: msdos                         <==分割表形式
Number  Start   End     Size    Type      File system  Flags
 1      32.3kB  107MB   107MB   primary   ext3         boot
 2      107MB   10.6GB  10.5GB  primary   ext3
 3      10.6GB  15.8GB  5240MB  primary   ext3
 4      15.8GB  41.2GB  25.3GB  extended
 5      15.8GB  16.9GB  1045MB  logical   linux-swap
 6      16.9GB  18.9GB  2056MB  logical   ext3
 7      18.9GB  19.2GB  263MB   logical   linux-swap
[  1 ]  [  2 ]  [  3  ] [  4  ] [  5  ]   [  6  ]
```

1. Number：這個就是分割槽的號碼啦！舉例來說，1號代表的是 /dev/hdc1 的意思；
2. Start：起始的磁柱位置在這顆磁碟的多少 MB 處？有趣吧！它以容量作為單位喔！
3. End：結束的磁柱位置在這顆磁碟的多少 MB 處？
4. Size：由上述兩者的分析，得到這個分割槽有多少容量；
5. Type：就是分割槽的類型，有primary, extended, logical等類型；
6. File system：就如同 fdisk 的 System ID 之意。

EXT2

- Ext2 stands for second extended file system.
- It was introduced in 1993. Developed by Rémy Card.
- This was developed to overcome the limitation of the original ext file system.
- Ext2 does not have journaling feature.
- On flash drives, usb drives, ext2 is recommended, as it doesn't need to do the over head of journaling.
- Maximum individual file size can be from 16 GB to 2 TB
- Overall ext2 file system size can be from 2 TB to 32 TB

EXT3

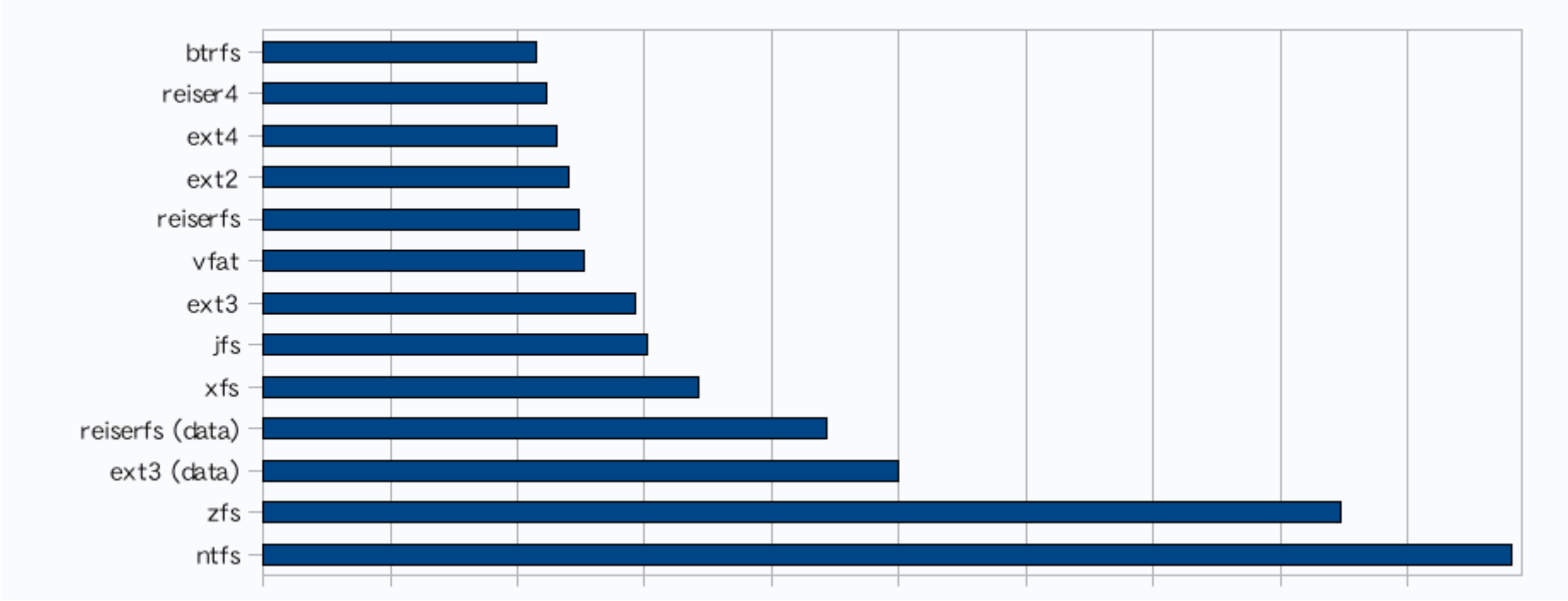
- Ext3 stands for third extended file system.
- It was introduced in 2001. Developed by Stephen Tweedie.
- Starting from Linux Kernel 2.4.15 ext3 was available.
- The main benefit of ext3 is that it allows journaling.
- Journaling has a dedicated area in the file system, where all the changes are tracked. When the system crashes, the possibility of file system corruption is less because of journaling.
- Maximum individual file size can be from 16 GB to 2 TB
- Overall ext3 file system size can be from 2 TB to 32 TB
- There are three types of journaling available in ext3 file system.
  - Journal – Metadata and content are saved in the journal.
  - Ordered – Only metadata is saved in the journal. Metadata are journaled only after writing the content to disk. This is the default.
  - Writeback – Only metadata is saved in the journal. Metadata might be journaled either before or after the content is written to the disk.
- You can convert a ext2 file system to ext3 file system directly (without backup/restore).

EXT4

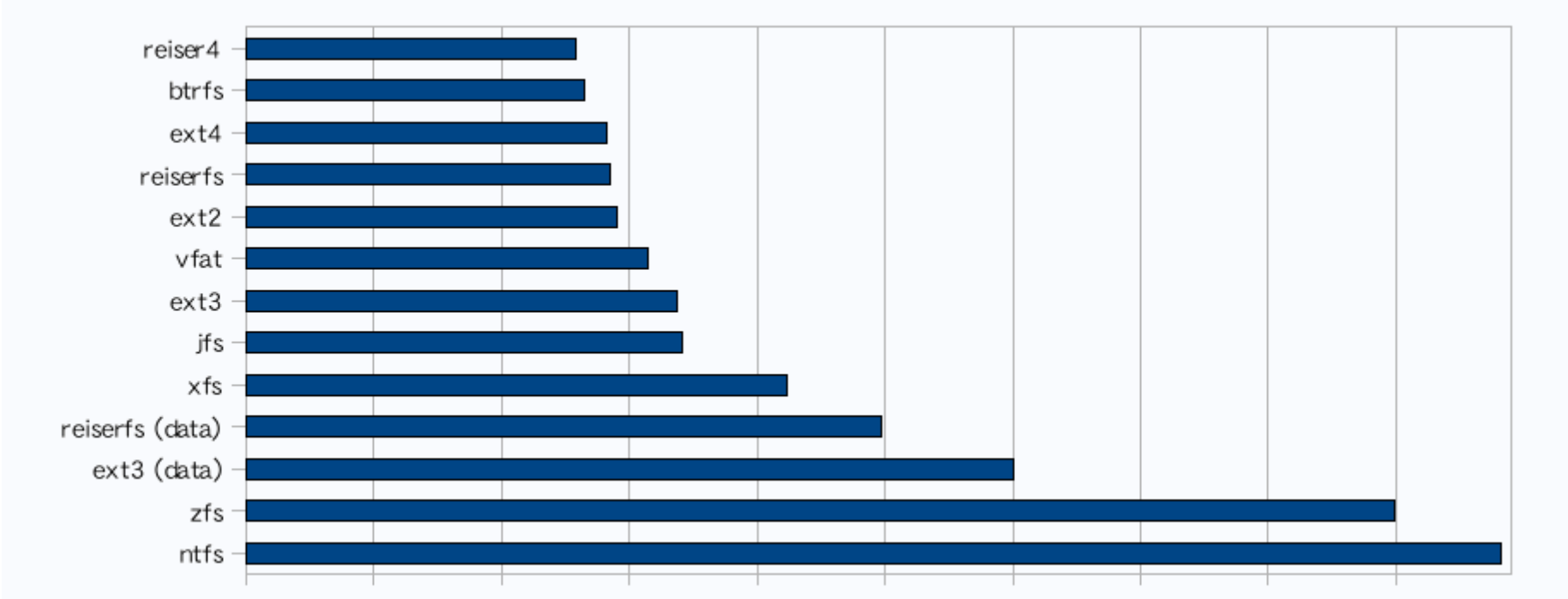
- Ext4 stands for fourth extended file system.
- It was introduced in 2008.
- Starting from Linux Kernel 2.6.19 ext4 was available.
- Supports huge individual file size and overall file system size.
- Maximum individual file size can be from 16 GB to 16 TB
- Overall maximum ext4 file system size is 1 EB (exabyte). 1 EB = 1024 PB (petabyte). 1 PB = 1024 TB (terabyte).
- Directory can contain a maximum of 64,000 subdirectories (as opposed to 32,000 in ext3)
- You can also mount an existing ext3 fs as ext4 fs (without having to upgrade it).
- Several other new features are introduced in ext4: multiblock allocation, delayed allocation, journal checksum. fast fsck, etc. All you need to know is that these new features have improved the performance and reliability of the filesystem when compared to ext3.
- In ext4, you also have the option of turning the journaling feature "off".

- ext2：老牌 Linux 檔案系統，不支援 journaling。
- ext3：當今各大 Linux 預設使用的檔案系統。支援 journaling。
- ext3 (data)：加上 journal_data 功能的 ext3。
- ext4：ext3 的下一版本。已正式進入 kernel 2.6.28 中。
- reiserfs：號稱最快的 FS。Linux 上第一個支援 journaling 的檔案系統。
- reiserfs (data)：加上 journal_data 功能的 reiserfs。
- reiser4：reiserfs 的下一版。（尚未進入 kernel 中）
- jfs：由 IBM 所開發的 journaling 型檔案系統。已停止開發。
- xfs：由 SGI 所開發的 journaling 型檔案系統。
- vfat：古老 DOS/Windows 檔案系統，不支援 journaling。
- ntfs：現今 Windows 的主流檔案系統。在 Linux 上是經由 fuse 來支援 ntfs。
- zfs：由 Sun 所開發的終極檔案系統。在 Linux 上是經由 fuse 來支援 zfs。
- btrfs：下一代 Linux 預設使用的檔案系統。已進入 kernel 2.6.29 RC1 的測試分支中。

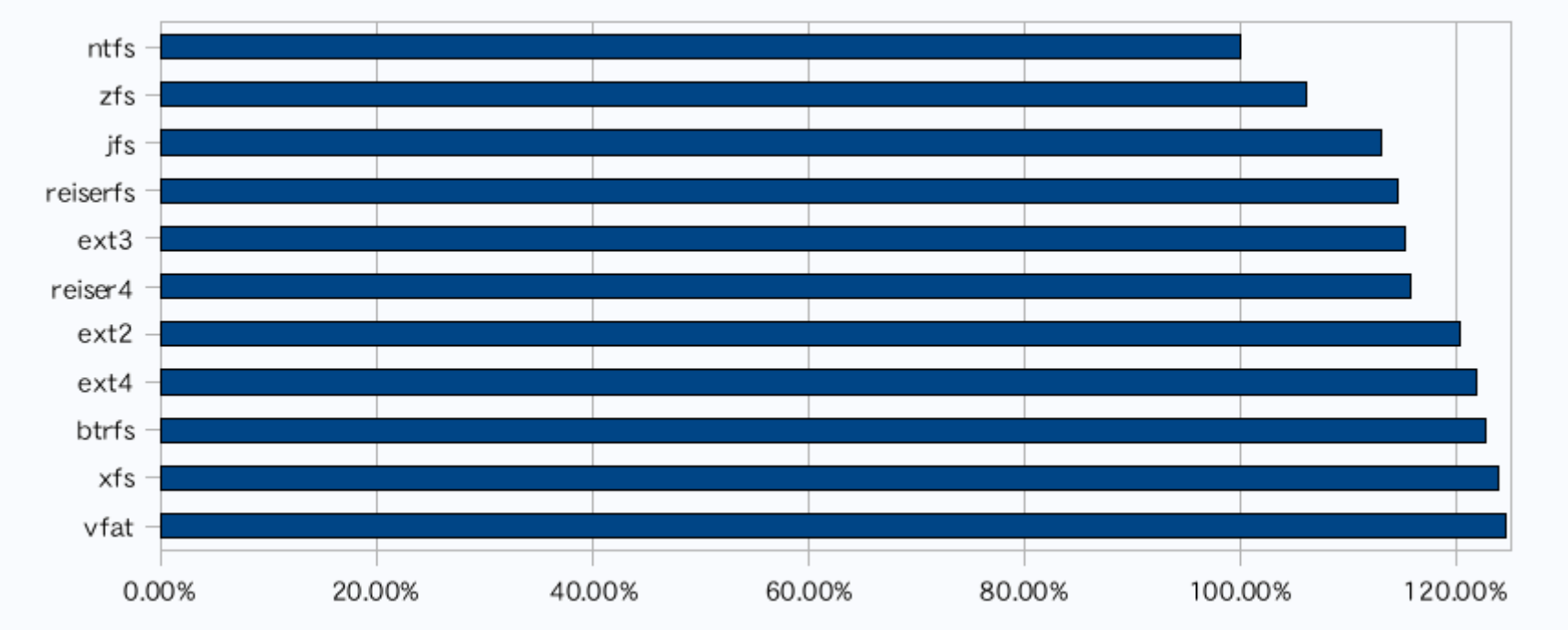
剛開始寫入硬碟的速度：



在磁碟裡的資料散亂後，寫入硬碟的速度：



FRAGMENT 的比率：



讀出、寫入、列出、搜尋、計算剩餘空間等等的效能表現：