Large Disk mini-HOWTO 中譯版

作者: Andries Brouwer, *jaeb@cwi.nl*
譯者: Asd L. Chen, *asdchen@ms1.hinet.net*
v1.0, 26 June 1996 翻譯日期: 10-13 November 1997

所有有關 disk geometry 及 1024 cylinder 的限制．

目錄

1 問題所在

假如你的磁碟超過 1024 個磁簇(cylinders)． 還有, 假如你的作業系統使用基本輸出入系統(BIOS)．那麼你會遇到一個問題, 因為一般磁碟輸出入輸入所使用的 INT13 BIOS 介面以一個 10 位元(bit) 的欄位來操作磁簇, 所以無法存取第 1024 及之後的磁簇．

幸運的是, Linux 不使用 BIOS, 所以沒有問題．

話是這麼說, 但有兩件事例外:

(1) 當你在啓動系統時, Linux 還沒開始執行所以無法讓你避免這個問題．這對 LILO 以及類似的啓動載入程式(boot loaders)有些影響．

(2) 使用磁碟的所有作業系統必須同意分割區的位置．換句話說, 如果你在一顆磁碟上使用 Linux 以及, 例如 DOS, 那麼兩者必須以相同的方式解讀分割表的資料．這對 Linux 核心以及 fdisk 有些影響.

底下是對所有相關細節更詳細的描述．注意, 我使用 2.0.8 版核心原始程式做爲參考．其它的版本可能有一點點出入．

2 啓動

當系統啓動時, BIOS 從第一個磁碟(或從軟碟)讀取磁區 0 (一般通稱的 MBR - Master Boot Record, 主啓動磁區)並跳至在該處的程式碼 - 通常是一些啓動載入程式(bootstrap loader)． 這些小小的啓動程式一般不會有自己的磁碟驅動程式而會使用 BIOS 所提供的服務．這意謂著只有整個 Linux 核心都位於開頭的 1024 個磁簇內時才能夠被啓動．

這個問題很容易解決: 確定核心(也許還包括其它啓動時用到的檔案, 像是 LILO map 檔) 是放在一個 BIOS 可以存取的到, 全都在開頭的 1024 個磁簇內的分割區裡 - 這可以(可能)是第一個或第二個磁碟．

另一點是啓動載入程式與 BIOS 必須同意彼此對磁碟邏輯(geometry)上的看法． 給 LILO ‘`linear`’ 這個選項參數可能會有些幫助．細節後述．

3 磁碟 geometry 以及分割區

如果你的磁碟上有好幾種作業系統, 每一種使用一個或多個分割區．那麼對於分割區位於何處不同的看法可能導致災難性的後果．

MBR 中包含一個分割表描述分割區(主分割區: primary) 在那裡．有四個表格給四個主要分割區使用, 它們看起來像

```
struct partition {
        char active;    /* 0x80: bootable, 0: not bootable */
        char begin[3];  /* CHS for first sector */
        char type;
        char end[3];    /* CHS for last sector */
        int start;      /* 32 bit sector number (counting from 0) */
        int length;     /* 32 bit number of sectors */
};
```

(其中 CHS 是磁簇磁頭磁區: Cylinder/Head/Sector 的縮寫)

因此, 有項資訊是重覆的: 分割區的位置可以由 24 位元的 `begin` 以及 `end` 欄位, 和 32 位元的 `start` 以及 `length` 欄位給定．

Linux 只使用 `start` 以及 `length` 欄位, 故最多可以處理包含 2^32 個磁區的分割區, 也就是, 最大 2 TB 的分割區．這是現今磁碟機的兩百倍, 所以也許足夠往後十年的需求．

不幸的是, BIOS INT13 呼叫使用三個位元組的 CHS 編碼, 10 個位元作爲磁簇號碼, 8 個位元作爲磁頭號碼, 及 6 個位元作爲磁軌上的磁區號碼． 可能的磁簇號碼是 0-1023, 可能的磁頭號碼是 0-255, 而磁軌上可能的磁區號碼爲 1-63(是的, 磁軌上的磁區是由 1 起算, 不是 0)． 以這 24 位元最多可以定址 8455716864 個位元組(7.875 GB), 這是 1983 年磁碟機的兩百倍．

更不幸的是, 標準的 IDE 介面容許 256 個磁區磁軌, 65536 個磁簇以及 16 個磁頭．它自己本身可以存取 2^37 = 137438953472 個位元組(128 GB), 但是加上 BIOS 方面 63 個磁區與 1024 個磁簇的限制後只剩 528482304 個位元組(504 MB)可以定址的到．

這不足以應付現今的磁碟, 人們使用各種硬體或軟體上的方法來克服．

4 轉換與磁碟管理程式

沒有人對磁碟的'眞實' geometry 有興趣．磁軌的磁區數通常是變動的 - 接近磁碟外圍的磁軌有比較多的磁區 - 所以沒有'眞實'的每磁軌磁區數． 對於使用者而言最好是把磁碟當作編號 0,1,..., 的磁區組合成的線性陣列, 讓控制器去找出磁區究竟位於磁碟的那裡．

此線性編號一般通稱爲 LBA．對於 geometry 爲 (C,H,S) 的磁碟而言屬(c,h,s) 的線性位址爲 c*H*S+h*S+(s-1)．所有 SCSI 控制器都使用 LBA, 某些 IDE 控制器也是．

如果 BIOS 把這 24 個位元(c,h,s) 轉換成 LBA 並餵給懂得 LBA 的控制器, 那麼又可以定址到 7.875 GB．並不足以應付所有的磁碟, 但仍然是個改進．注意此處 BIOS 使用的 CHS, 它不再與'實體'有任何關係．

當控制器不懂何爲 LBA 但是 BIOS 知道如何轉換時有些類似的方法可行．(在 BIOS 設定中通常稱爲 'Large'．)現在 BIOS 將呈現 geometry 爲(C',H',S')給作業系統, 而在與磁碟控制器溝通時則使用(C,H,S)．通常 S=S', C'=C/N 而 H'=H*N, 其中 N 是確保 C'<=1024 之 2 的最小次方(所以 C'=C/N 時捨去的數浪費少許容量)．再一次, 這允許存取最多達 7.875 GB．

如果 BIOS 不知道 'Large' 或是 'LBA', 那麼還是有軟體的解決方案．像是 OnTracker 或 EZ-Drive 這些個磁碟管理程式會以它們自己的函式(routines)替換掉 BIOS 的． 通常這是藉由將磁碟管理程式放在 MBR 及其後幾個磁區(OnTrack 稱這些程式碼爲 DDO: Dynamic Drive Overlay)來達成的, 所以它會在任何其它作業系統之前被啓動． 這也就是爲什麼在安裝磁碟管理程式後從軟碟啓動可能會出問題．

這影響可能多於或少於 BIOS 轉換 - 但特別是在相同的磁碟上跑數種不同的作業系統時, 磁碟管理程式可能引起許多問題．

Linux 從 1.3.14 版開始支援 OnTrack 磁碟管理程式, 從 1.3.29 開始 支援 EZ-Drive ．下面有些更進一步的資訊．

5 核心的 IDE 磁碟轉換

如果 Linux 偵測到 IDE 磁碟上有某些磁碟管理程式存在, 它將會試著使用與該磁碟管理程式相同的方式來重新對應磁碟, 所以 Linux 看到與, 例如 DOS 配合 OnTrack 或是 EZ-Drive 相同的磁碟分割． 然而, 當你在指令列上指定 geometry 時, 就不會做任何的重新對應 - 所以一行 '`hd=`*cyls*`,`*heads*`,`*secs*' 指令列選項可能取消掉與磁碟管理程式的相容．

此重新對應的方式是嘗試 4,8,16,32,64,128,255 磁頭數(H*C 保持不變)直到 C <= 1024 或是 H = 255．

細節如下 - 小節的抬頭是出現在相對應之啓動訊息裡的字串．在此以及在這份文件任何其它地方中分割的型態都以十六進位數字表示．

5.1 EZD

偵測到 EZ-Drive , 因爲第一個主要分割區型態爲 55 ．如上述重新對應 geometry, 且忽略從第 0 磁區讀入的分割表 - 以第 1 磁區的分割表取代．磁碟的區塊號碼(block numbers) 沒有改變, 但對磁區 0 的寫入會轉向磁區 1．此動作可以藉由修改在 *ide.c* 中的

```
#define FAKE_FDISK_FOR_EZDRIVE 0
```

並重新編譯核心來改變．

5.2 DM6:DDO

偵測到 OnTrack DiskManager(在第一個磁碟上), 因爲第一個主要分割區型態爲 54 ·如上述重新對應 geometry 而且整個磁碟平移 63 個磁區· (所以舊的磁區 63 變成磁區 0)然後從新的第 0 磁區讀入新的 MBR (與分割表)·此平移當然是爲 DDO 留空間 - 這也就是爲什麼其它磁碟不必平移·

5.3 DM6:AUX

偵測到 OnTrack DiskManager(在其它磁碟上), 因爲第一個主要分割區型態爲 51 或 53·如上述重新對應 geometry ·

5.4 DM6:MBR

偵測到某舊版的 OnTrack DiskManager, 並非藉由分割區型態, 而是簽名(signature)·(測試在 MBR 裡第 2,3 位元組的偏移值是否不大於 430, 而且在此偏移位址找到的 short 等於 0x55AA, 後面並跟著一個奇數的位元組) 再一次如上述重新對應 geometry ·

5.5 PTBL

最後, 核心會嘗試從主分割區的 `start` 以及 `end` 值推斷轉換方式: 若某些分割區的 `start` 以及 `end` 磁簇小於 256, 而且 `start` 以及 `end` 磁區號碼分別爲 1 和 63 , 而且 end 磁頭爲 31, 63, 或 127, 那麼, 因爲依慣例分割區會在磁簇邊界結束, 而且更因爲 IDE 介面最多使用 16 個磁頭, 故推測有開啓 BIOS 轉換, 分別使用 32, 64 或 128 磁頭數重新對應 geometry· (也許這裡有點瑕疵, *genhd.c* 不應該測試磁簇號碼前兩個位元嗎?)無論如何, 當目前的 geometry 已經爲每磁軌 63 個磁區且至少這麼多磁頭時, 不會做重新對應 (因爲這可能意謂著重對應已完成)·

6 結論

這到底意謂著什麼? 對 Linux 的使用者而言只有一件事: 就是他們必須得確保 LILO 以及 fdisk 使用正確的 geometry , 其中'正確'的定義對 fdisk 而言是與其它在同一個磁碟上的作業系統所使用的 geometry 相同, 而對 LILO 而言是能夠在啓動時期成功地與 BIOS 交談的 geometry·(這兩者通常相符·)

fdisk 如何得知該 geometry ? 它詢問核心, 使用 `HDIO_GETGEO` ioctl·但使用者可以交談式地或在指令列上重定 geometry·

LILO 如何得知該 geometry ? 它詢問核心, 使用 `HDIO_GETGEO` ioctl· 但使用者可以用 '`disk=`' 選項重定 ·也可以給 LILO 一個 `linear` 選項, 如此它將在其 map 檔中儲存 LBA 位址以取代 CHS 位址, 並且在啓動時期找出該 geometry 來使用(藉由 INT 13 功能呼叫 8 來詢問磁碟的 geometry)·

核心如何知道該怎麼回答? 首先, 使用者可能以 '`hd=`*cyls*`,`*heads*`,`*secs*' 指令列選項明確地指定 geometry, 否則核心將詢問硬體·

6.1 IDE 細節

讓我詳細說明·IDE 驅動程式有四個關於 geometry 的資訊來源·第一個(G_user)是使用者在指令列上所指定的· 第二個(G_bios)是 BIOS 的固定磁碟參數表(只用於第一及第二個磁碟), 在系統啓動時, 切換

至 32 位元模式之前讀入· 第三個(G_phys)及第四個(G_log) 是由 IDE 控制器傳回, 作爲對 IDENTIFY 指令的回應 - 它們是 '實體的' 以及 '目前邏輯上的' geometries·

另一方面, 對於 geometry 驅動程式需要兩個值: 其中之一是 G_fdisk, 由 `HDIO_GETGEO` ioctl 傳回, 另一個是 G_used , 這是執行輸出入時實際使用的· 如果給定 G_user 則 G-fdisk 以及 G_used 兩者都會設爲 G_user, 當此資訊是根據 CMOS 所提供時則設爲 G_bios , 其它情形設爲 G_phys·如果 G_log 看起來合理則 G_used 就設爲 G_log· 不然, 如果 G_used 不合理而 G_phys 看起來合理那麼 G_used 就設爲 G_phys·此處的'合理'代表磁頭數在 1-16 的範圍内·

換個方式說: 指令列選項大於 BIOS , 並且決定 fdisk 看到的樣子, 但如果它指定轉換的 geometry(磁頭數大於 16), 則核心會藉由 IDENTIFY 指令的輸出重定它·

6.2 SCSI 細節

在 SCSI 方面情況有一點點不同, 因爲 SCSI 指令已經使用邏輯區塊號碼, 所以 'geometry' 對實際的輸出入完全沒關係· 然而, 分割區的格式仍然是相同的, 所以 fdisk 必須得捏造些 geometry , 並且也在此使用 `HDIO_GETGEO` - 眞的, fdisk 不會分辨 IDE 以及 SCSI 磁碟· 你可以從下面的詳細描述見到各種驅動程式捏造一些個不同的 geometry ·眞是, 一團混亂·

如果你沒有使用 DOS 或這類系統, 那麼避免使用所有額外的轉換設定, 可能的話, 儘管使用 64 磁頭, 每磁軌 32 磁區 (良好的, 方便每磁簇 1 MB), 如此當你把磁碟從一個控制器換到另一個去時不會遇到任何問題· 某些 SCSI 磁碟驅動程式 (aha152x,pas16,ppa,qlogicfas,qlogicisp)非常在意與 DOS 的相容性而不允許只有 Linux 的系統使用超過 8 GB 的容量, 這是隻臭蟲·

眞實的 geometry 是什麼? 最簡單的答案是沒有這種東西·如果眞有的話, 你不會想知道, 而且的的確確從不, 永不需告訴 fdisk 或是 LILO 或核心有關它的事·這絕對是 SCSI 控制器與磁碟之間的事· 讓我重覆這句話: 只有蠢蛋會告訴 fdisk/LILO/Kernel SCSI 磁碟眞實的 geometry ·

但如果你好學且堅持, 可以問磁碟機自己·有個重要指令 READ CAPACITY 將會傳回磁碟的總容量, 而且有個 MODE SENSE 指令 Rigid Disk Drive Page(page 04) 會傳回磁簇以及磁頭的數目(這是不能改變的資訊), 而在 Format Page(page 03)有每磁區的位元組, 以及每磁軌的磁區數· 這數字一般與 notch 有關, 而且每磁軌的磁區數是變動的 - 外圍的磁軌擁有比内圈磁軌多的磁區·Linux 程式 scsiinfo 會給予這項資訊· 其中有許多繁瑣的細節, 而且很明白的, 沒有人(也許甚至是作業系統)需要使用這項資訊· 還有, 因爲我們只關心 fdisk 以及 LILO , 一般得到的回答像 C/H/S=4476/27/171 - 這樣的值 fdisk 根本不能使用, 因爲分割表只保留 10resp. 8resp. 6 bits 給 C/H/S·

那核心之 `HDIO_GETGEO` 從何處取得其資訊? 嗯, 不是從 SCSI 控制器, 就是推論猜測· 有些驅動程式似乎認爲我們想知道 '眞相' , 但我們當然只想知道 DOS 或 OS/2 FDISK (或 Adaptec AFDISK 等等)所用的·

注意, Linux fdisk 需要磁頭數 H 以及每磁軌磁區數 S 以便轉換 LBA 磁區號碼成爲 c/h/s 位址, 但磁簇數 C 在此轉換中並未扮演什麼角色· 有些驅動程式使用 (C,H,S) = (1023,255,63) 來表示磁碟容量至少爲 1023*255*63 個磁區·這是不幸的, 因爲這不能顯示實際的大小, 而且將限制大部份版本之 fdisk 的使用者其磁碟最大到 8 GB - 現今實際的限制·

在下面的描述中, M 表示磁碟的全部容量, 而 C,H,S 是磁簇, 磁頭以及每磁軌磁區數·如果我們把 C 當作 M/(H*S) 那給 H,S 就可以滿足·

依預設, H=64, S=32.

aha1740, dtc, g_NCR5380, t128, wd7000:

H=64, S=32.

aha152x, pas16, ppa, qlogicfas, qlogicisp:

H=64, S=32 除非 C > 1024, 此情況下 H=255, S=63, C = min(1023, M/(H*S)). (故 C 被截斷, 且 H*S*C 不是磁碟容量的近似值. 這將會混搖淆大部份版本的 fdisk.) *ppa.c* 程式碼使用 M+1 取代 M 並認爲這是因爲在 *sd.c* 裡的一隻臭蟲使 M 的值少一・

advansys:

H=64, S=32 除非 C > 1024 而且還開啓 BIOS 中的 '> 1 GB' 選項, 此情況下 H=255, S=63.

aha1542:

詢問控制器使用兩種可能的 schemes 中的那一種, 並且使用 H=255, S=63 或 H=64, S=32. 前者有個啓動訊息 "aha1542.c: Using extended bios translation".

aic7xxx:

H=64, S=32 除非 C > 1024, 而且還給了 "extended" 啓動參數, 或在 SEEPROM 或 BIOS 設了 'extended' 位元, 此情況下 H=255, S=63.

buslogic:

H=64, S=32 除非 C >= 1024, 而且還啓動控制器的擴充轉換, 此情況下若 M < 2^22 則 H=128, S=32; 否則 H=255, S=63. 然而, 選擇 (C,H,S) 之後, 讀入分割表, 若三種可能的 (H,S) = (64,32), (128,32), (255,63) 中 endH=H-1 的值看來可行則使用該對 (H,S) , 並印出啓動訊息 "Adopting Geometry from Partition Table".

fdomain:

從 BIOS Drive Parameter Table 找出 geometry 資訊, 或從分割表讀取並使用 H=endH+1, S=endS 給第一個分割區, 若非空, 或使用 H=64, S=32 for M < 2^21 (1 GB), H=128, S=63 for M < 63*2^17 (3.9 GB) and H=255, S=63 otherwise.

in2000:

使用 (H,S) = (64,32), (64,63), (128,63), (255,63) 中第一個讓 C <= 1024 的. 此情況下, 在 1023 截掉 C .

seagate:

從磁碟讀取 C,H,S. (眞誠實!) 如果 C 或 S 太大, 放入 S=17, H=2 並倍增 H 直到 C <= 1024. 這表示 H 將爲 0 如果 M > 128*1024*17 (1.1 GB). 這是隻臭蟲.

ultrastor and u14_34f:

三種對應之一 ((H,S) = (16,63), (64,32), (64,63)) 根據控制器的對應模式而定.

如果驅動程式沒有指定 geometry , 我們就回到使用分割表或磁碟總容量來推斷猜測・

仔細看看分割表・因爲依慣例分割區在磁簇邊界結束, 我們可以爲任何分割區定 end=(endC,endH,ednS) , 只要放入 H = `endH+1` 及 S = `endS`・ (記得磁區由 1 起算・)更明確地的說・如果有個不是空的分割區 , 則以最大的 `beginC` 計・ 對於此分割區, 看看 `end+1`, 計算加上 `start` 以及 `length` 並且假設此分割區在某磁簇邊界結束・ 如果兩個值都相符, 或 `endC` = 1023 且 `start+length` 是 `(endH+1)*endS` 的倍數, 那麼假定此分割區眞的是在磁簇邊界, 並放入 H = `endH+1` 以及 S = `endS`・ 如果不對, 不是因爲沒有分割區, 就是因爲它們的大小很奇怪, 那麼只看磁碟容量 M・ 演算法: 放入 H = M/(62*1024)(無條件進位),S = M/(1024*H)(無條件進位), C = M/(H*S)(無條件捨去)・這能產生一 (C,H,S) 其中 C 最大 1024 而 S 最大 62・