# Large Disk HOWTO

Andries Brouwer, `aeb@cwi.nl` v2.2m, 15 Febbraio 2000

Notizie fondamentali sulla geometria dei dischi e sul limite dei 1024 cilindri. Traduzione di Gianluca Vezzù, *vezzu@tin.it* .

## 1 La problematica

Facciamo l'ipotesi di avere un disco che abbia più di 1024 cilindri. Ipotizziamo inoltre di avere un sistema operativo che utilizzi la vecchia interfaccia all'I/O del disco fornita dall'INT13. Avremo allora un problema perché tale interfaccia per indirizzare le operazioni di I/O su di un generico cilindro impiega un campo a 10 bit, pertanto i cilindri dal 1024 in poi sono inaccessibili.

Per fortuna Linux non utilizza le chiamate al BIOS, eliminando così ogni eventuale problema alla radice.

Purtroppo le cose non vanno bene in due casi:

(1) Quando fate il boot Linux non sta ancora girando e quindi non può preservarvi dai problemi derivanti dal BIOS. Ciò ha delle ripercussioni per LILO e per analoghi boot loader.

(2) È necessario che tutti i sistemi operativi installati sullo stesso disco concordino sulla locazione delle partizioni. In altri termini, se utilizzate, sia Linux che, per esempio, DOS, entrambi i sistemi operativi devono interpretare la tabella delle partizioni nello stesso modo. Ciò comporta delle conseguenze per il kernel di Linux e per `fdisk`.

Appresso si riporta una descrizione dettagliata di tutti gli argomenti più importanti. Prestate attenzione al fatto che come riferimento ho utilizzato la versione 2.0.8 del kernel. Le altre versioni possono avere delle leggere differenze.

## 2 Sommario

Hai appena acquistato un nuovo disco. Che fare adesso? Bene, dal punto di vista software usi `fdisk` (o meglio ancora `cfdisk`) per creare le partizioni, poi `mke2fs` per creare il filesystem, e per finire `mount` per attaccare il nuovo filesystem a quello esistente.

*Circa un anno fa potevo scrivere:* Non hai bisogno di leggere questo HOWTO perché *non* ci sono più problemi con gli hard disk di grandi dimensioni attuali. La maggior parte di quelli che sembrano essere problemi è dovuta agli utenti che pensano che ci possa essere un problema e quindi installano un gestore dei dischi, o entrano nella modalità avanzata di `fdisk`, o specificano esplicitamente la geometria del disco a LILO o la passano al kernel da riga di comando.

Tuttavia, le tipologie di problemi più comuni sono: (i) hardware obsoleto, (ii) presenza di sistemi operativi sullo stesso disco e qualche volta (iii) la fase di boot.

*Attualmente la situazione è peggiorata.* Probabilmente la versione 2.3.21 e le succesive permetteranno di nuovo la gestione di tutti i tipi di dischi.

Consigli:

Dischi SCSI di grandi dimensioni: Linux li supporta sin dalle prime versioni. Non è richiesta nessuna azione specifica.

Dischi IDE di grandi dimensioni (superiori agli 8.4 GB): procurati un kernel stabile recente (2.0.34 o superiore). Di norma tutto dovrebbe procedere bene, specialmente se sarete così saggi da non chiedere al BIOS la traslazione del disco con la LBA o con accorgimenti simili.

Dischi IDE di grandissime dimensioni (superiori ai 33.8 GB): vai alla sezione 12.1 (Problemi dei controller IDE con dischi di dimensioni oltre i 34 GB). Se LILO si blocca in fase di boot specifica l'opzione `5.1 (linear)` nel file di configurazione `/etc/lilo.conf`.

Ci sono dei problemi legati alla geometria che possono essere risolti fornendo esplicitamente la geometria al kernel/LILO/fdisk.

Se utilizzate una versione datata di `fdisk` e vi dà un avviso di 6 (overlapping) partitions: ignorate i messaggi, oppure controllate con `cfdisk` che sia tutto a posto.

Se pensate che la dimensione del vostro disco sia sbagliata fate attenzione a non confondere le 3 (unità di misura) binarie con quelle decimali e tenete presente che lo spazio libero che `df` riporta in un disco vuoto è di qualche percento più piccolo della dimensione della partizione perché c'è dello spazio riservato alle funzionalità di amministrazione.

Ora se pensate ancora che ci siano dei problemi o se più semplicemente siete curiosi proseguite nella lettura.

# 3 Unità di Misura e Dimensioni

Un kilobyte (kB) corrisponde a 1000 byte. Un megabyte (MB) corrisponde a 1000 kB. Un gigabyte (GB) corrisponde a 1000 MB. Un terabyte (TB) corrisponde a 1000 GB. Questa è la definizione delle unità di misura nel *sistema internazionale (SI)* . Tuttavia ci sono persone che considerano 1 MB=1024000 byte e parlano di dischetti da 1.44 MB e altre che pensano che 1 MB=1048576 byte. Nel presente documento seguirò *lo standard attuale*

e scriverò Ki, Mi, Gi, Ti per indicare le unità binarie, così che i floppy hanno dimensioni di 1440 KiB (1.47 MB, 1.41 MiB), 1 MiB sono 1048576 byte (1.05 MB), 1 GiB sono 1073741824 byte (1.07 GB) e 1 TiB sono 1099511627776 byte (1.1 TB).

I produttori di hard disk seguono il sistema SI utilizzando quindi la notazione decimale. I messaggi in fase di avvio di Linux e qualche programma come fdisk utilizzano i simboli MB e GB ad indicare l'utilizzo della notazione binaria o l'utilizzo di una notazione mista binaria/decimale. Prima di dire d'avere un disco più piccolo di quanto dichiarato dal costruttore calcolate la sua dimensione nelle unità decimali (o più semplicemente in byte).

A proposito della terminologia e delle abbreviazioni adottate per specificare le unità binarie, *Knuth*

ha suggerito una notazione alternativa,nello specifico propone di utilizzare KKB, MMB, GGB, TTB, PPB, EEB, ZZB, YYB e di definirli come *kilobyte esteso*, *megabyte esteso*, ... *yottabyte esteso*. Lui stesso ha scritto: 'Prestate attenzione al fatto che raddoppiando le lettere si identifica sia la natura binaria che quella di grande dimensione'. Questa è una buona proposta - 'gigabyte esteso' suona meglio di 'gibibyte'. Per i nostri scopi la sola cosa importante è sottolineare che un megabyte contiene 1000000 di byte e che è necessario utilizzare qualche altro termine ed abbreviazione se si sottointende qualcosa di diverso.

## 3.1 Dimensione dei Settori

In questa trattazione si assume che un settore abbia dimensione pari a 512 byte. Questa assunzione è quasi sempre vera, ma, per esempio, certi dischi MO (MagnetoOttici) utilizzano settori di 2048 byte. Quindi tutte le capacità elencate sopra devono essere moltiplicate per quattro (quando utilizzate `fdisk` su questi dischi controllate di avere la versione 2.9i o superiore e date l'opzione '-b 2048).

### 3.2 Dimensione del Disco

Un disco con C cilindri, H testine e S settori per traccia ha C*H*S settori totali e può memorizzare C*H*S*512 byte. Per esempio, se l'etichetta del disco riporta C/H/S=4092/16/63 allora il disco ha 4092*16*63=4124736 settori e può contenere 4124736*512=2111864832 byte (2.11 GB). C'è un'accordo industriale per assegnare ai dischi più grandi di 8.4 GB le dimensioni C/H/S=16383/16/63, così la dimensione del disco non può più essere letta dalla terna di valori C/H/S riportata dal disco.

## 4 Accesso al Disco

Per poter leggere o scrivere qualsiasi cosa su un disco occorre specificare una posizione sul disco stesso per esempio indicando il numero del settore o del blocco. Se il disco è SCSI il numero del settore viene passato al controller SCSI ed è capito dal disco. Se il disco è un IDE che utilizza la modalità LBA si ha la stessa cosa. Ma se il disco è obsoleto, RLL o MFM o IDE senza supporto LBA allora l'hardware del disco si aspetta una terna (cilindro,testina,settore) per individuare il punto desiderato sul disco.

La corrispondenza tra la numerazione lineare e la terna di tre cifre è di seguito mostrata. Per un disco con C cilindri, H testine e S settori/traccia la posizione indicata dalla notazione come terna o CHS (c,h,s) è la stessa di quella indicata dalla notazione lineare o LBA c*H*S + h*S + (s-1) (la sottrazione di una unità è legata al fatto che nella notazione come terna i settori sono contati a partire da 1 e non da 0).

Concludendo possiamo dire che per accedere ad un disco non SCSI obsoleto è necessario conoscerne la sua *geometria* ossia i valori C, H e S.

### 4.1 Accesso BIOS al Disco e il Limite dei 1024 cilindri

Al contrario di altri sistemi Linux non utilizza il BIOS. Il BIOS, antecedente all'LBA, fornisce le routine di I/O su disco attraverso l'INT13 che prevedono come ingresso la terna (c,h,s) (più precisamente: `AH` seleziona la funzione da utilizzare, `CH` contiene gli 8 bit bassi del numero dei cilindri, `CL` contiene nei bit 7-6 i due bit alti del numero dei cilindri e nei bit 5-0 il numero del settore, `DH` contiene il numero delle testine e `DL` contiene il numero identificativo del drive (80h or 81h). Questo spiega una parte dello schema della tavola delle partizioni).

Noi abbiamo la terna CHS codificata su tre byte in cui 10 bit sono per il numero dei cilindri, 8 per le testine e 6 per il numero dei settori (numerati da 1 a 63). Da ciò risulta come il numero dei cilindri possa variare da 0 a 1023 e come il BIOS non sia in grado di indirizzare più di 1024 cilindri.

I programmi per DOS e Windows non sono stati modificati quando sono stati introdotti i dischi IDE con il supporto LBA così sia il DOS che Windows continuano ad aver bisogno della geometria del disco solo per poter dialogare con il BIOS, pur non essendo necessaria per le operazioni di I/O. Questo significa che Linux ha bisogno di conoscere la geometria del disco in quei sistemi ove sia richiesto il dialogo tra BIOS e altri sistemi operativi presenti, anche con un disco attuale.

Questi problemi sono iniziati più o meno circa quattro anni fa, quando apparvero dischi che non potevano essere indirizzati dalle funzioni dell'INT13 (perché i 10+8+6=24 bits della terna (c,h,s) non possono indirizzare più di 8.5 GB) e fu quindi progettata una nuova interfaccia per il BIOS: la cosidetta INT13 Estesa dove DS:SI punta ad un Disk Address Packet di 16 byte che contiene un numero assoluto di inizio blocco di 8 byte.

Molto lentamente il mondo Microsoft sta traghettando verso l'utilizzo delle funzioni fornite dall'INT13 Esteso. Probabilmente fra pochi anni nessun sistema moderno equipaggiato con hardware moderno necessiterà più del concetto di "geometria del disco".

## 4.2 Storia del BIOS e dei limiti dell'IDE

**Specifiche ATA (per dischi IDE) - il limite dei 137 GB**

65536 cilindri (numerati da 0-65535), 16 testine (numerate da 0-15), 255 settori/traccia (numerati da 1-255), corrispondono a 267386880 settori (di 512 byte ciascuno) che equivalgono ad un massimo di 136902082560 byte (137 GB). Questo non è un problema attuale (1999) ma lo diverrà tra qualche anno.

**BIOS Int 13 - il limite degli 8.5 GB**

1024 cilindri (numerati da 0-1023), 256 testine (numerate da 0-255), 63 settori/traccia (numerati da 1-63), corrispondono a 8455716864 byte (8.5 GB). Questo è un limite molto gravoso ai nostri giorni perché significa che il DOS non può utilizzare i dischi di grosse dimensioni attuali.

**Il limite dei 528 MB**

Se gli stessi valori c,h,s sono utilizzati dalla chiamata all'INT 13 del BIOS e dal controller I/O IDE entrambe le limitazioni si sovrappongono permettendo l'accesso al massimo a 1024 cilindri, 16 testine, 63 settori/traccia per una capacità totale di 528482304 byte (528MB), l'infame limite dei 504 MiB del DOS con i vecchi BIOS. Questo problema si è sentito a partire dal 1993 circa e gli utenti sono ricorsi ai più svariati trucchi sia hardware (LBA) sia firmware (traslazione del BIOS) sia software (gestori dei dischi). Il concetto di 'traslazione' è stato introdotto nel 1994: il BIOS può usare una geometria per dialogare con il disco e un'altra, contraffatta, per dialogare con il DOS ed effettuare la conversione tra le due.

**Il limite dei 2.1 GB (Aprile 1996)**

Alcuni BIOS datati allocano solo 12 bit nella CMOS RAM per memorizzare il numero dei cilindri. Come conseguenza il valore massimo rappresentabile è 4095. Da ciò deriva che sono indirizzabili solamente 4095*16*63*512=2113413120 byte. Se si ha un disco più grande si avrà un blocco del sistema in fase di avvio. Questo ha reso i dischi con la geometria 4092/16/63 abbastanza diffusi. Ancor'oggi molti dischi di grandi dimensioni hanno un jumper per fornire la geometria 4092/16/63 Per ulteriori informazioni: *over2gb.htm* . *Altri BIOS*

non si bloccano ma rilevano un disco molto più piccolo, ad esempio 429 MB invece di 2.5 MB.

**Il limite dei 3.2 GB**

I BIOS Phoenix 4.03 e 4.04 avevano un baco che causava il blocco del sistema quando si impostavano nel setup della CMOS dischi con capacità superiori ai 3277 MB. Vedi: *over3gb.htm* .

**Il limite dei 4.2 GB (Feb 1997)**

La traslazione effettuata dal BIOS (ECHS= CHS Estesa, detta anche 'Supporto ai dischi di grandi dimensioni' o semplicemente 'Large') ricorsivamente raddoppia il numero delle testine e contemporaneamente dimezza il numero dei cilindri passati al DOS finché il numero dei cilindri è al massimo 1024. Il DOS e Windows95 non possono gestire 256 testine, e nel caso abbastanza comune in cui il disco fornisce 16 testine ciò significa che questo meccanismo è utilizzabile per gestire al massimo 8192*16*63*512=4227858432 byte (con una geometria contraffatta di 1024 cilindri, 128 testine e 63 settori/traccia). È da osservare che ECHS non modifica il numero dei settori per traccia, così se non sono 63 la capacità gestibile sarà ancora più bassa. Vedi: *over4gb.htm* .

**Il limite dei 7.9 GB**

Un po' furbescamente alcuni BIOS aggirano il problema precedente fissando a 15 il numero di testine ('ECHS rivisto') in modo da poter ottenere una geometria contraffatta con 240 testine. Sono indirizzabili 1024*240*63*512=7927234560 byte.

**Il limite degli 8.4 GB**

Se il BIOS è in grado di utilizzare 255 testine e 63 settori/traccia ('LBA assistita' o più semplicemente 'LBA') può indirizzare 1024*255*63*512=8422686720 byte, un po' meno del limite precedente di 8.5 GB questo perché le geometrie con 256 testine sono da evitarsi (la traslazione utilizza come numero di testine H il primo valore della sequenza 16, 32, 64, 128, 255 per il quale la capacità totale sia minore od eguale a 1024*H*63*512, quindi calcola il numero dei cilindri C come la capacità totale diviso per (H*63*512)).

**Il limite dei 33.8 GB (Agosto 1999)**

Ci sono ulteriori difficoltà con dischi di dimensioni superiori ai 33.8 GB. Il problema sta nel fatto che i valori predefiniti di 16 testine e 63 settori/traccia corrispondoo ad un numero di cilindri maggiore di 65535, quantità che non è rappresentabile da una variabile di tipo short. Attualmente molti BIOS non sono in grado di gestire tali unità(vedi per esempio: *Aggiornamenti Asus*

per trovare versioni aggiornate del BIOS che supportino tali unità). I kernel precedenti le versioni 2.2.14 / 2.3.21 devono essere aggiornati. Vedi 12.1 (Problemi dei controller IDE con dischi di dimensioni superiori ai 34 GB) più sotto.

Per ulteriore materiale su questo argomento vedi: *Breaking the Barriers* ("Rompere le barriere"), e per ulteriori dettagli *IDE Hard Drive Capacity Barriers* ("Limiti della capacità degli HD IDE").

I dischi superiori agli 8.4 GB riportano la loro geometria come 16383/16/63. Ciò significa che la 'geometria' è obsoleta e che la capacità totale del disco non può più essere calcolata dalla geometria.

# 5 Avvio del sistema

Quando il sistema viene inizializzato il BIOS legge il settore 0 (definito come MBR - Master Boot Record) dal primo disco fisso (o dal floppy) e lancia il programma che vi trova - di solito un bootstrap loader. Il loader residente nel settore 0 non ha i suoi driver per cui utilizza i servizi del BIOS. Ciò significa che il kernel di Linux deve essere interamente contenuto nei primi 1024 cilindri per essere caricato.

Questo problema si risolve molto semplicemente verificando che il kernel (e gli altri file utilizzati durante il boot, come le mappe dei file di LILO) sia in una partizione contenuta intieramente nei primi 1024 cilindri del disco in modo che il BIOS vi possa accedere - probabilmente questo significa utilizzare il primo o il secondo disco.

Create quindi una piccola partizione, diciamo di 10 MB, così c'è spazio per più di un kernel, facendo attenzione che sia contenuta entro i primi 1024 cilindri del primo o del secondo disco. Montatela in `/boot` così LILO vi metterà i file di cui necessita.

## 5.1 LILO e l'opzione 'linear'

Un altro problema è che sia il boot loader che il BIOS devono vedere la stessa geometria del disco. LILO inoltra al kernel la richiesta di informazioni circa la geometria ma molte volte gli autori dei driver dei dischi hanno la brutta abitudine di utilizzare la tavola delle partizioni per desumere la geometria del disco invece di specificare a LILO qual è la geometria utilizzata dal BIOS. Così facendo spesso la geometria fornita dal BIOS non è corretta. In questi casi può essere conveniente passare a LILO l'opzione '`linear`' in modo che non necessitando dei parametri della geometria durante la fase di inizializzazione del sistema (memorizza gli indirizzi lineari nelle mappe) faccia la conversione in indirizzi lineari all'avvio. Perché questa non è un'opzione predefinita? Bene, occorre tener presente che il suo uso comporta un problema. LILO con l'opzione 'linear' attivata non è in grado di conoscere il numero dei cilindri come conseguenza non può generare nessun avvertimento se una parte del kernel si trova oltre il limite dei 1024 cilindri. Alla fine vi potreste trovare con un sistema che non è in grado di avviarsi.

### 5.2 Un "bug" di LILO

Le versioni di LILO inferiori alla v21 hanno un difetto: la conversione degli indirizzi effettuata durante la fase di avvio ha un "bug": quando il prodotto c*H è maggiore od eguale a 65536 si hanno degli errori di overflow durante il calcolo. Per valori di H superiori a 64 si ha un limite più stretto sui valori attribuibili a c rispetto al solito c < 1024; per esempio, con H=255 e con una versione datata di LILO si deve avere c < 258 (c=cilindro dove risiede l'immagine del kernel, H=numero delle testine del disco).

### 5.3 1024 cilindri non sono 1024 cilindri

Tim Williams scrive: 'Ho la mia partizione Linux nei primi 1024 cilindri e tuttavia non c'è verso di di avviarla. Quando l'ho ridimensionata a meno di 1 GB le cose sono andate a posto.' Cosa può essere? Bene, il disco in parola è un disco SCSI con un controller AHA2940UW che usa sia H=64, S=32 (ossia cilindri di 1 MiB = 1.05 MB) che H=255, S=63 (ossia cilindri di 8.2 MB) a seconda delle opzioni impostate nel firmware e nel BIOS. Senza dubbio il BIOS assume la prima e trova il limite dei 1024 settori in corrispondenza di una capacità di 1 GiB mentre Linux utilizza la seconda e LILO trova il limite agli 8.4 GB.

## 6 Geometria dei dischi fissi, delle partizioni e 'sovrapposizione'

Se utilizzate diversi sistemi operativi sul vostro disco fisso ognuno avrà a disposizione una o più partizioni. La posizione delle partizioni deve essere univoca per tutti i sistemi presenti onde evitare delle conseguenze catastrofiche.

Il MBR contiene la tavola delle partizioni che descrive la posizione delle partizioni primarie. Nella tavola ci sono 4 campi per le 4 partizioni primarie, ogni campo è descritto da una struttura del tipo

```
struct partizione {
  char attiva;       /* 0x80: avviabile; 0: non avviabile */
  char inizio[3];    /* CHS del primo settore */
  char tipo;
  char fine[3];      /* CHS dell'ultimo settore */
  int partenza;      /* numero identificativo del settore a 32 bit
                        (si conta a partire da 0) */
  int lunghezza;     /* numero totale dei settori a 32 bit */
};
```

(dove CHS significa Cilindri/Testine/Settori - Cylinder/Head/Sector).

la struttura dà informazioni ridondanti. La posizione di una partizione si ricava dai campi `inizio` e `fine` entrambi a 24 bit e dai campi partenza e `lunghezza` questi ultimi a 32 bit.

Linux utilizza solo i campi `inizio` e `lunghezza`, può quindi gestire partizioni che abbiano meno di 2^32 settori (circa 2TiB). Questa dimensione è sessanta volte più grande dei dischi attualmente disponibili, probabilmente sarà sufficiente per i prossimi otto anni e forse oltre.

(Le partizioni possono essere molto grandi tuttavia c'è un limite alla dimensione massima di un singolo file che nei sistemi a 32 bit non può essere più grande di 2GiB.)

Il DOS utlizza i campi `inizio` e `fine` e la chiamata all'INT13 del BIOS per accedere al disco, può quindi indirizzare dischi non più grandi di 8.4 GB pur effettuando la traslazione (le partizioni non possono superare i 2.1 GB perché occorre tener conto delle restrizioni imposte al filesystem dalla FAT16). Lo stesso dicasi per Windows 3.11 e WfWG e Windows NT 3.*.

Windows 95 ha il supporto per l'interfaccia all'INT13 Esteso e utilizza un tipo speciale di partizione (c, e, f invece di b, 6, 5) per indicare che tale partizione può essere accessibile in questo modo. Quando si utilizzano questi tipi di partizione i campi `inizio` e `fine` contengono delle informazioni fasulle (1023/255/63). Windows 95 OSR2 ha introdotto il fylesystem FAT32 (partizioni di tipo b o c) che permette partizioni di dimensioni al massimo di 2 TiB.

A cosa sono dovute le stupidaggini che apprendiamo da `fdisk` circa la 'sovrapposizione' delle partizioni quando in effetti non c'è nulla di sbagliato? Bene - qualche volta c'è un 'errore': se date un'occhiata ai campi `inizio` e `fine` di tali partizioni, come fa il DOS, queste si sovrappongono (tale 'errore' non può essere corretto perché i campi non possono memorizzare un numero di cilindri superiore a 1024 - ci sarà sempre 'sovrapposizione' non appena avrete più di 1024 cilindri). Tuttavia, se voi date uno sguardo ai campi `inizio` e `lunghezza`, come fa Linux, e come fa anche Windows 95 nel caso di partizioni di tipo c, e o f allora tutto procede per il meglio. Concludendo potete ignorare questi avvertimenti quando usate `cfdisk` e sul vostro disco è installato solo Linux. Occorre prestare attenzione quando il disco è condiviso con il DOS. Usate il comando `cfdisk -Ps /dev/hdx` e `cfdisk -Pt /dev/hdx` per controllare la tabella delle partizioni del disco `/dev/hdx`.

# 7 Traduzione e Gestori dei Dischi

La geometria del disco (a testine, cilindri e tracce) è qualcosa che ci arriva dal tempo di MFM e RLL. In quel periodo la geometria corrispondeva ad una realtà fisica. Oggigiorno, con le interfacce IDE o SCSI, nessuno è più interessato a conoscere la geometria 'reale' del disco. Il numero di settori per traccia è variabile - ci sono più settori per traccia nelle zone più esterne del disco - cosicché non esiste un numero 'reale' di settori per traccia. Anzi è l'esatto opposto: il comando IDE INITIALIZE DRIVE PARAMETERS (91h) serve per richiedere al disco quante testine e settori per traccia pensa di avere. È abbastanza comune che i dischi di grandi dimensioni che hanno due testine comunichino al BIOS 15 o 16 testine mentre il BIOS ne riporta 255 all'utente.

Per l'utente è meglio considerare un disco alla stregua di un vettore di settori numerati progressivamente 0,1,...., e lasciare al controller del disco il compito di localizzare i settori. La numerazione progressiva è definita LBA.

Si riporta adesso il funzionamento logico. Il DOS, o qualsiasi boot loader, dialoga con il BIOS utilizzando la notazione (c,h,s). Il BIOS converte la (c,h,s) nella notazione LBA utilizzando la falsa geometria che l'utente sta utilizzando. Se il disco accetta la LBA allora la utilizza nelle operazioni di I/O su disco. Altrimenti, il BIOS la riconverte in (c',h',s') utilizzando la geometria falsa in uso e questi nuovi valori sono impiegati nell'I/O su disco.

È da rimarcare che c'è un po' di confusione nell'utilizzo dell'espressione 'LBA': come termine che descrive le possibilità di un disco significa 'Linear Block Addressing' (al contrario dell'indirizzamento CHS), mentre come termine presente nel setup del BIOS descrive uno schema di traslazione che qualche volta viene definito come 'assisted LBA' - cfr più sotto 4.2 (Linux e il limite degli 8 GiB dei controller IDE).

Se il controller non supporta la LBA ma è il BIOS ad effettuare la traduzione si ha un comportamento simile a quanto descritto sopra (nel setup del BIOS è spesso indicata come modalità 'Large'). In questo caso il BIOS passa al sistema operativo la geometria (C',H',S') e utilizza (C,H,S) per dialogare con il controller del disco. Di norma si ha che: S = S', C' = C/N e H' = H*N, dove N rappresenta la più piccola potenza di due che assicura la validità della diseguaglianza C' <= 1024 (si spreca spazio a causa dell'arrotondamento dovuto al rapporto C' = C/N). Anche in questo caso si possono indirizzare più di 8.4 GB (7.8 GiB).

(La terza opzione del setup è di solito 'Normal', che non effettua nessuna traslazione).

Se il BIOS non supporta né la modalità 'Large' né la 'LBA' si deve ricorrere a delle soluzioni software. Gestori del disco ("Disk Manager") come OnTrack o EZ-Drive sostituiscono con le loro routine di gestione del disco quelle del BIOS. Di solito vengono installati nell'MBR e nei settori seguenti (OnTrack definisce questi programmi come DDO: Dynamic Drive Overlay) in modo che vengono eseguiti prima del sistema operativo. Per questo motivo quando si inizializza il sistema da dischetto ci possono essere problemi.

I risultati che si ottengono sono più o meno gli stessi di quelli forniti da un BIOS che effettua la traduzione - ma, quando sono presenti più sistemi operativi sullo stesso disco ci possono essere molti problemi derivanti dall'utilizzo dei disk manager.

Linux è compatibile con OnTrack Disk Manager dalla versione 1.3.14 e con EZ-Drive da quella 1.3.29. Altri dettagli sono riportati appresso.

# 8 Traduzione dei dischi IDE operata dal kernel

Se il kernel di Linux rileva la presenza di un disk manager su un disco IDE cerca di rimappare il disco nello stesso modo in cui l'ha mappato il disk manager, così vede le stesse partizioni che ad esempio il DOS gestisce con OnTrack o EZ-Drive. Tuttavia se si specifica la geometria da riga di comando NON viene effettuata la rimappatura del disco. Una riga di comando del tipo '`hd=`*cilindri*`,`*testine*`,`*settori*' fa perdere ogni compatibilità con il disk manager.

Se tutto questo vi dà fastidio e se conoscete qualcuno che può ricompilarvi un nuovo kernel cercate il file `linux/drivers/block/ide.c` e cancellate dalla funzione `ide_xlate_1024()` il test: `if (drive->forced_geom) {--;return 0;}`.

La rimappatura si fa, posto il prodotto H*C costante, variando il valore del numero di testine (4, 8, 16, 32, 64, 128, 255) finché si verifica C <= 1024 o H = 255.

Il titolo dei paragrafi che seguono corrisponde alle stringhe che appaiono in fase di boot quando Linux rileva la presenza di un disk manager. I tipi delle partizioni si devono intendere espressi in notazione esadecimale.

## 8.1 EZD

EZ-Drive è rilevato perché assegna alla partizione primaria numero uno il tipo 55. La geometria è rimappata come detto sopra. La tabella delle partizioni presente nel settore 0 non viene presa in considerazione perché viene letta quella presente nel settore 1. Il numero dei blocchi del disco non è cambiato, l'operazione di scrittura sul settore 0 è reindirizzata al settore 1. Questo comportamento può essere modificato ricompilando il kernel con: `#define FAKE_FDISK_FOR_EZDRIVE 0` in `ide.c`.

## 8.2 DM6:DDO

OnTrack DiskManager (sul primo disco) è rilevato perché assegna alla partizione primaria numero uno il tipo 54. La geometria è rimappata come detto sopra e tutto il disco è traslato di 63 settori (così facendo il vecchio settore 63 diventa il nuovo settore 0). Dopo questa operazione sul settore 0 viene letto un nuovo MBR (con la relativa tavola della partizioni). Naturalmente la traslazione dei settori è necessaria per creare lo spazio per il DDO, questo spiega perché negli altri dischi non si applica tale operazione.

### 8.3 DM6:AUX

OnTrack DiskManager (sugli altri dischi) è rilevato perché assegna alla partizione primaria numero uno il tipo 51 o 53. La geometria è rimappata come detto sopra.

### 8.4 DM6:MBR

Questa è una vecchia versione di OnTrack DiskManager rilevata da una firma e non da un assegnamento di tipo ad una partizione (si verifica che l'offset trovato nel 2 e 3 byte dell' MBR non sia maggiore di 430 e che lo short trovato in questo offset sia eguale a 0x55AA, e inoltre che sia seguito da un byte di disparità). La geometria, come nei casi precedenti, è rimappata come detto sopra.

### 8.5 PTBL

Si possono dedurre le informazioni necessarie alla traduzione del disco utilizzando un test che verifica l'`inizio` e la `fine` della partizione primaria. Se l'`inizio` e la `fine` di una partizione sono nei settori numero 1 e 63, rispettivamente e se le testine finali sono 31, 63, 127 o 254 allora, considerato che è prassi terminare le partizioni entro le dimensioni di un cilindro, e dacché l'interfaccia IDE utilizza al massimo 16 testine, si può presupporre che il BIOS abbia effettuato una traduzione e che la geometria del disco sia stata rimappata per utilizzare 32, 64, 128 o 255 testine, rispettivamente. Quando la geometria rilevata ha già 63 settori per traccia e altrettante testine non viene effettuata la rimappatura (perché questo significa che probabilmente ne è già stata fatta una).

## 9 Conclusioni

Cosa significa tutto ciò? Per gli utenti Linux una sola cosa: che devono verificare che LILO e `fdisk` utilizzino la geometria corretta. Per `fdisk` la geometria 'corretta' è quella utilizzata dagli altri sistemi operativi installati sullo stesso disco mentre per LILO è quella che permette un'interazione senza errori con il BIOS in fase di inizializzazione (di solito i due aspetti coincidono).

Come rileva la geometria `fdisk`? Interroga il kernel, utilizzando la funzione ioctl.h `HDIO_GETGEO`. L'utente è tuttavia in grado di forzare la geometria in modo interattivo o da linea di comando.

Come rileva la geometria LILO? Interroga il kernel utilizzando la ioctl `HDIO_GETGEO`. Tuttavia l'utente può forzare la geometria utilizzando l'opzione '`disk=`' nel file `/etc/lilo.conf` (cfr lilo.conf(5)). È possibile fornire a LILO l'opzione `linear` in modo da memorizzare nella mappa dei file i valori della LBA invece dei CHS e recupererà la geometria da utilizzare in fase di avvio mediante la funzione 8 dell'INT 13 per richiedere la geometria del disco.

Com'è in grado di rispondere il kernel? Bene, prima di tutto, l'utente può aver specificato una geometria esplicita fornendo le opzioni '`hda=`*cyls*`,`*heads*`,`*secs*' al kernel da riga di comando (cfr. bootparam(7)), probabilmtente di persona od indicando al "boot loader" di fornire tali valori al kernel. Per esempio, si può indicare a LILO di fornire tali informazioni opzionali aggiungendo una riga del tipo '`append = "hda=`*cyls*`,`*heads*`,`*secs*`"`' in `/etc/lilo.conf` (cfr. lilo.conf(5)). D'altrocanto il kernel è in grado di congetturare una geometria utilizzando possibilmente i valori ottenuti dal BIOS o dai dispositivi hardware.

È possibile (a partire dalla versione 2.1.79) cambiare i valori della geometria congetturata dal kernel utilizzando il filesystem `/proc`. Per esempio:

```
# sfdisk -g /dev/hdc
/dev/hdc: 4441 cylinders, 255 heads, 63 sectors/track
# cd /proc/ide/ide1/hdc
```

```
# echo bios_cyl:17418 bios_head:128 bios_sect:32 > settings
# sfdisk -g /dev/hdc
/dev/hdc: 17418 cylinders, 128 heads, 32 sectors/track
#
```

## 9.1 Calcolo dei parametri di LILO

Qualche volta è utile impostare una precisa geometria da riga di comando: '`hda=`*cyls*`,`*heads*`,`*secs*'. Quasi sempre si vuole un valore di *secs*=63, lo scopo di fornire tale valore è quello di specificare il valore *heads* (valori ragionevoli in questo periodo sono *heads*=16 e *heads*=255). Quali valori si potrebbero specificare per *cyls*? Quei valori che forniranno la capacità totale corretta di C*H*S settori. Per esempio, per un drive con 71346240 settori (36529274880 byte) si potrebbe calcolare C come 71346240/(255*63)=4441 (usando, e.g., il programma `bc`) e fornire quindi come parametri di avvio: `hdc=4441,255,63`. Come si può stabilire la capacità totale corretta? Per esempio:

```
# hdparm -g /dev/hdc | grep sectors
 geometry     = 4441/255/63, sectors = 71346240, start = 0
# hdparm -i /dev/hdc | grep LBAsects
 CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=71346240
```

indica due modi per stabilire il numero totale si settori pari a 71346240. I messaggi del kernel:

```
# dmesg | grep hdc
...
hdc: Maxtor 93652U8, 34837MB w/2048kB Cache, CHS=70780/16/63
 hdc: [PTBL] [4441/255/63] hdc1 hdc2 hdc3! hdc4 < hdc5 > ...
```

riportano (perlomeno) 34837*2048=71346176 settori e 70780*16*63=71346240 settori (almeno). In questo caso il secondo valore è quello corretto ma in generale entrambi sono arrotondati per difetto. Questo è un buon modo per definire in maniera approssimata le dimensioni del disco quando `hdparm` non è disponibile. Non assegnate mai a *cyls* un valore troppo grande! Nel caso di dischi SCSI il numero preciso dei settori è indicato dai messaggi forniti dal kernel all'avvio:

```
SCSI device sda: hdwr sector= 512 bytes. Sectors= 17755792 [8669 MB] [8.7 GB]
```

(MB e GB sono arrotondati, non per difetto, e in 'notazione binaria').

# 10 Dettagli

## 10.1 Dettagli sui dischi IDE - le sette geometrie

Il driver IDE può determinare la geometria di un disco in quattro modi. Il primo (G_user) è quello di passare le informazioni da linea di comando. Il secondo (G_bios) è la lettura in fase di inizializzazione della Tavola dei Parametri del Disco nel BIOS (solo per il primo e il secondo disco) prima del passaggio alla modalità a 32 bit. Il terzo (G_phys) e il quarto (G_log) sono le risposte del controller IDE al comando IDENTIFY, sono rispettivamente la geometria 'fisica' e 'logica' del disco.

Il driver ha bisogno di due valori per determinare la geometria: uno è dato da G_fdisk, ottenuto da una chiamata a `HDIO_GETGEO` ioctl, l'altro da G_used, che è utilizzato per le operazioni di I/O. Sia G_fdisk che G_used sono inizializzati da G_user se vengono passati i valori, altrimenti da G_bios o da G_phys. Se G_log ha un valore ragionevole questo viene assegnato a G_used. Nel caso in cui G_used non sia ragionevole ma lo sia G_phys quest'ultimo valore sarà assegnato a G_used. Per 'valore ragionevole' si intende un numero di testine compreso tra 1 e 16.

Per dirla in altri termini: i parametri passati da linea di comando forzano i valori del BIOS e determinano la geometria che vedrà `fdisk`. Tuttavia se i parametri si riferiscono ad una geometria tradotta con più di 16 testine l'I/O del kernel sarà forzato dai valori ricavati dal comando IDENTIFY.

Bisogna osservare che il G_bios è piuttosto inattendibile: per i sistemi inizializzati da un driver SCSI il primo ed il secondo disco possono essere SCSI e la geometria che il BIOS attribuisce a sda è quella usata dal kernel per hda. Tuttavia, i dischi che non sono impostati nel Setup del BIOS non possono essere visti da quest'ultimo. Ciò significa che, per esempio, in un sistema IDE dove non sia presente hdb le geometrie riportate dal BIOS per il primo e per il secondo disco siano attribuite a hda e hdc.

## 10.2 Dettagli sui dischi SCSI

Per i dischi SCSI la situazione è diversa. Questi dischi utilizzano la LBA per cui determinare una 'geometria' è del tutto irrilevante. Il formato della tavole delle partizioni è praticamente lo stesso, `fdisk` deve crearsi una qualche geometria, utilizza `HDIO_GETGEO`, perché non è in grado di distinguere tra dischi IDE e SCSI. Come si vedrà in seguito ogni driver crea una sua geometria. Veramente una grande confusione.

Se non si utilizza il DOS è meglio evitare le impostazioni di traduzione estese ed utilizzare se possibile 64 testine, 32 settori per traccia (per gli esigenti può andar bene 1MB per cilindro) così facendo si evitano problemi se si cambia controller. Alcuni driver (aha152x, pas16, ppa, qlogicfas, qlogicisp) sono così legati ai limiti del DOS da non permettere l'utilizzo di più di 8GiB anche nel caso in cui Linux sia il solo sistema installato. Questo è un baco.

Qual è la geometria reale? La risposta più facile è che non esiste. E se anche esistesse voi non dovreste occuparvene e di sicuro MAI E POI MAI dovreste richiedere tali informazioni a `fdisk`, a LILO o al kernel. Questa è una cosa di cui di occupano il controller SCSI e il disco. Lasciatemelo ripetere: solo gli stupidi richiedono a `fdisk`/LILO/kernel la geometria di un disco SCSI.

Tuttavia se siete persone curiose ed insistenti potete leggere tali valori dal disco stesso. Il comando READ CAPACITY dà la capacità totale del disco; MODE SENSE riporta il numero di cilindri e testine nella Rigid Disk Drive Geometry Page (page 04, questi valori non possono essere modificati) e il numero di byte per settore e i settori per traccia nella Format Page (page 03). Quest'ultimo valore dipende dalla formattazione ed è variabile, le tracce più esterne hanno più settori di quelle interne. Il programma `scsiinfo` fornisce questi dati. Ci sono molti altri dettagli e difficoltà da superare, è palese che nessuno utilizzerà questo genere di informazioni (molto probabilmente nemmeno lo stesso sistema operativo). Considerato che stiamo parlando di `fdisk` e di LILO i valori tipici sono: C/H/S=4476/27/171, valori questi che non possono essere utilizzati da `fdisk` perché la tabella delle partizioni riserva per la terna C/H/S rispettivamente 18, 8 e 6 bit.

Ma allora come fa `HDIO_GETGEO` a ricavare tali valori? Interrogando o il controller SCSI o facendo delle richieste appropriate al kernel. Qualche driver sembra pensare che le nostre richieste vogliano determinare la geometria reale mentre a noi interessa conoscere il valore dei parametri utilizzati da FDISK sia che si utilizzi DOS che OS/2 (o Adaptec AFDISK, ecc...)

È da sottolineare che `fdisk` utilizza il numero delle testine H e dei settori per traccia S per convertire i numeri dei settori LBA nel corrispondente indirizzo c/h/s; il numero dei cilindri C non riveste un ruolo importante in questa conversione. Certi driver indicano che la capacità del drive è di almeno 1023*255*63 settori assegnando a C,H,S i seguenti valori:1023,255,63. Così facendo non si ottiene la vera dimensione del disco e si limita l'accesso ai primi 8GB a molte versioni di `fdisk` - un vero problema d'attualità.

Nelle descrizioni a seguire con M si indica la capacità totale del disco, con C,H,S il numero di cilindri, testine e settori per traccia. È sufficente ottenere i valori di H e S se si ricava C dalla relazione M / (H*S).

I valori predefiniti di H e S sono rispettivamente 64 e 32.

**aha1740, dtc, g_NCR5380, t128, wd7000:**

H=64, S=32.

**aha152x, pas16, ppa, qlogicfas, qlogicisp:**

H=64, S=32 se C <= 1024, altrimenti H=255, S=63, C = min(1023, M/(H*S)). (C rappresenta un valore troncato, H*S*C non è un'approssimazione della capacità M. Per questo molte versioni di `fdisk` si comportano in modo anomalo). Il sorgente `ppa.c` utilizza M+1 invece di M per cui genera un messaggio d'errore dicendo che M è più piccolo di 1 a causa di un baco in `sd.c`.

**advansys:**

H=64, S=32 se C <= 1024, se è abilitata l'opzione '> 1 GB' nel BIOS i valori assunti sono rispettivamente 255 e 63.

**aha1542:**

Richiede al controller quale delle due traduzioni possibili sta utilizzando e utilizza o H=255, S=63 o H=64, S=32. Nel primo caso durante il boot si ha il messaggio "aha1542.c: Using extended bios translation".

**aic7xxx:**

H=64, S=32 se C <= 1024, inoltre se viene utilizzato il parametro "extended" o si imposta il bit 'extended' nella SEEPROM o nel BIOS i valori saranno: H=255, S=63. L'opzione extended viene sempre impostata se non viene rilevata nessuna SEEPROM con il kernel 2.0.36 mentre con la versione 2.2.6 se non viene trovata la SEEPROM l'opzione è impostata solo se l'utente specifica in fase d'avvio il parametro "extended" (nel caso venga rilevata una SEEPROM tale parametro verrà ignorato). Tutto ciò significa che un sistema che funziona con il 2.0.36 può non avviarsi con il 2.2.6 (e richiedere l'opzione 'linear' per LILO o il parametro 'aic7xxx=extended' da passare al kernel al boot)

**buslogic:**

H=64, S=32 finché C >= 1024. Se si imposta la traduzione estesa nel controller si hanno due casi a seconda che sia verificata o meno la condizione M < 2^22: nel primo caso H=128, S=32 altrimenti H=255, S=63. Dopo aver effettuato la scelta della terna (C,H,S) viene letta la tabella delle partizioni e si verifica che il valore endH=H-1 appartenga ad una delle tre condizioni possibili (H,S) = (64,32), (128,32), (255,63). Sarà utilizzata la coppia che soddisfa la condizione, tale scelta è segnalata da un messaggio durante il boot ("Adopting Geometry from Partition Table").

**fdomain:**

Ricava le informazioni sulla geometria o dalla tabella dei parametri BIOS del drive o dalla tavola delle partizioni. Utilizza i valori: H=endH+1, S=endS per la prima partizione, purché non sia vuota, oppure H=64, S=32 se M < 2^21 (1 GB), H=128, S=63 se M < 63*2^17 (3.9 GB) e H=255, S=63 in tutti gli altri casi.

**in2000:**

Utilizza la prima coppia di valori (H,S) = (64,32), (64,63), (128,63), (255,63) che verifica la condizione C <= 1024. Nell'ultimo caso il valore di C è troncato a 1023.

**seagate:**

Legge C,H,S dal disco (orrore!). Se C o S sono troppo grandi allora imposta S=17 e H=2 e raddoppia H finché C <= 1024. Significa che H sarà impostato a zero nel caso in cui M > 128*1024*17 (1.1 GB). Questo è un baco.

**ultrastor and u14_34f:**

Utilizza una coppia di valori (H,S) in funzione di come il controller mappa il disco. ((H,S) = (16,63), (64,32), (64,63))

Se il driver non specifica la geometria del disco possiamo richiedere i dati alla tabella delle partizioni oppure utilizzare tutta la capacità del disco.

Diamo uno sguardo alla tabella delle partizioni. Per convenzione le partizioni terminano in un cilindro esterno per cui dato `fine` = (endC,endH,endS) per una partizione qualsiasi si pone H = `endH+1` e S = `endS` (ricordo che i settori si contano a partire da 1). Il procedimento è descritto appresso in modo più preciso. Tra le partizioni non vuote si considera quella con il valore `beginC` più grande. Si controlla il valore `end+1` di questa partizione, calcolato sia come somma di `partenza` con `lunghezza` sia ipotizzando che la partizione termini al confine di un cilindro. Se entrambi i valori coincidono, o se `endC` = 1023 e la somma `partenza+lunghezza` è multiplo di `(endH+1)*endS`, allora la partizione è allineata sul confine di un cilindro. Si pongono H = `endH+1` e S = `endS`. Nel caso in cui le verifiche siano negative o perché non ci sono partizioni o perché non hanno dimensioni note si utilizza la capacità del disco M. L'algoritmo è così schematizzabile: porre H = M/(62*1024) (arrotondato all'intero superiore), S = M/(1024*H) (arrotondato all'intero superiore), C = M/(H*S) (arrotondato all'intero inferiore). I valori di (C,H,S) che si ottengono sono al massimo 1024 per C e 62 per S.

# 11 Linux e il limite degli 8 GiB dei controller IDE

Il driver IDE di Linux ricava la geometria e la capacità di un disco (e molte altre cose) utilizzando una richiesta ATA IDENTIFY.

Fino a poco tempo fa il driver non accettava il valore di lba_capacity restituito se questo era maggiore del 10% rispetto alla capacità calcolata come prodotto di C*H*S. Ciò nonostante grazie ad accordi tra i produttori di dischi IDE di grandi dimensioni (quelli con più di 16514064 settori) forniscono i valori: C=16383, H=16, S=63 per un totale di 16514064 settori (7.8 GB) indipendentemente dalla loro dimensione reale che forniscono alla lba_capacity.

I kernel più recenti (2.0.34, 2.1.90) conoscono il problema e si comportano di conseguenza. Se avete un kernel datato che vede solamente i primi 8 GB di un disco più grande, e non volete aggiornarlo, provate a cambiare la funzione `lba_capacity_is_ok` in `/usr/src/linux/drivers/block/ide.c` come indicato:

```
static in lba_capacity_is_ok (struct hd_driveid *id) {
        id->cyls = id->lba_capacity / (id->heads * id->sectors);
        return 1;
}
```

Per fare un aggiornamento meno brutale utilizzate il kernel 2.1.90.

## 11.1 Complicazioni del BIOS

Come già detto i dischi di grandi dimensioni forniscono la geometria C=16383, H=16, S=63 indipendentemente dalle dimensioni reali, mentre la dimensione reale è indicata dal valore della LBAcapacity. Alcuni BIOS non la riconoscono, e traslano il 16383/16/63 in una terna con meno cilindri e più testine, per esempio 1024/255/63 o 1027/255/63. Così, il kernel non deve solo riconoscere la geometria 16383/16/63 ma anche quella generata da tali BIOS. Dal kernel 2.2.2 questa operazione di riconoscimento funziona in modo corretto (prendendo dal BIOS i valori H e S e calcolando C =capacità/(H*S)). Di norma questo problema si risolve impostando nei parametri del BIOS il disco come Normal (o, ancor meglio a None, non fornendo nessuna indicazione al BIOS). Se questa strada non è percorribile perché dovete fare il boot da questo disco oppure avete una partizione DOS/Windows, e non è possibile aggiornare la versione del kernel alla 2.2.2 o superiori, passate al kernel i parametri durante il boot.

Se il BIOS riporta la geometria 16320/16/63 ciò è fatto per ottenere dopo la traduzione la terna 1024/255/63.

Qui c'è un ulteriore problema. Se il disco è stato partizionato usando una geometria traslata il kernel, durante la fase di avvio, potrebbe vedere tale geometria utilizzata nella tabella delle partizioni e riportare: `hda: [PTBL] [1027/255/63]`. Questa è una brutta faccenda perchè ora il disco e di soli 8.4 GB. La versione 2.3.21 ha corretto questo problema tuttavia il passaggio dei parametri durante la fase di avvio può essere d'aiuto.

## 11.2 Impostare il numero delle testine per mezzo dei ponticelli (Jumper)

Molti dischi hanno dei ponticelli che permetto di selezionare una geometria a 15 o a 16 testine. La configurazione predefinita è quella a 16 testine. A volte entrambe le geometrie indirizzano lo stesso numero di settori altre quella a 15 testine ne indirizza un numero inferiore. C'è una buona ragione come spiega Petri Kaukasoina per spiegare queste due opzioni: 'Ho impostato un disco IBM Deskstar 16 GP (modello IBM-DTTA-351010) da 10.1 GiB per utilizzare 16 testine come da configurazione predefinita ma il mio vecchio PC (con BIOS AMI) non si avviava così ho dovuto spostare il ponticello sull'opzione che dà 15 testine. hdparm -i riporta RawCHS=16383/15/63 e LBAsects=19807200. Io utilizzo una configurazione 20960/15/63 per poter sfruttare tutta la capacità del disco.' Per ulteriori informazioni su come ponticellare tali dischi visitate il sito: *http://www.storage.ibm.com/techsup/hddtech/hddtech.htm* .

## 11.3 Ridurre la capacità totale di un disco mediante l'uso dei ponticelli

Molti dischi hanno dei ponticelli che permettono di mostrare le dimensioni del disco più piccole di quelle che sono. È un po' stupido a farsi e probabilmente nessun utente Linux vorrebbe mai utilizzare tale espediente ma alcuni BIOS non riescono a gestire i dischi di grandi dimensioni andando in "crash". La soluzione più comune è quella di non far vedere il disco all'avvio al BIOS, ma è possibile farlo solo se il disco non è quello di avvio.

Il primo limite grave era il limite di 4096 cilindri (che corrisponde, con 16 testine e 63 settori/traccia a 2,11 GB). Per esempio, il disco Fujitsu MPB3032ATU da 3.24 GB ha la geometria predefinita da 6704/15/63 ma può essere poticellato per fornire una geometria 4092/16/63 e riportare di conseguenza una LBAcapacity di 4124736 settori, in questo modo il sistema operativo non può congetturare che la dimensione reale è più grande. In questi casi (con un BIOS che si "schianta" se riconosce la reale dimensione del disco è necessario ricorrere al ponticello) è necessario informare Linux sulle dimensioni del disco fornendo i parametri all'avvio.

Questo è un caso sfortunato. Molti dischi possono essere "ponticellati" in modo da sembrare dischi da 2 GB e fornire quindi una geometria ridotta tipo 4092/16/63 o 4096/16/63 ma in grado di fornire il valore corretto della LBAcapacity. Tali dischi sono in grado di lavorare bene e di utilizzare la capacità totale con Linux indipendentemente dalle impostazioni dei ponticelli.

Un limite più recente è quello dei 12.1 (33.8 GB). I kernel di Linux precedenti alla versione 2.3.21 devono essere aggiornati per poter gestire dischi IDE di dimensioni superiori a queste. Alcuni dischi che superano tale limite possono essere ponticellati per sembrare dei dischi da 33.8 GB. Per esempio l'IBM Deskstar (DPTA-353750) da 37.5 GB può essere ponticellato per sembrare un disco da 33.8 GB e fornire la geometria 16383/16/63 come un qualsiasi altro disco di grandi dimensioni, ma la LBAcapacity di 66055248 (che corrisponde a 65531/16/63 o 4111/255/63). Sfortunatamente i ponticelli sembrano essere troppo efficaci - non influenzano solo ciò che il drive fornisce al sistema ma anche le operazioni di I/O: Petr Soucek comunica che questi parametri non sono d'aiuto nel caso di dischi d'avvio - ponticellando tali dischi ogni accesso ai settori oltre il settore 66055248 causa un errore di I/O. Tuttavia con schede madri che montino il BIOS Award 4.51PG tali dischi possono essere utilizzati come dischi d'avvio ed inoltre in tutta la loro capacità. Vedi anche: *the BIOS 33.8 GB limit*
.

# 12 Il limite dei 65535 cilindri in Linux

La chiamata `HDIO_GETGEO` memorizza il numero dei cilindri in una variabile di tipo short. Questo vuol dire che se si hanno più di 65535 cilindri il loro valore verrà troncato e (per una tipica configurazione SCSI com 1MiB di cilindri) un disco da 80 GiB apparirà come uno da 16 GiB. Una volta riconosciuto tale problema è facile evitarlo.

## 12.1 Problemi dei controller IDE con dischi di dimensioni superiori ai 34 GB

I dischi superiori ai 33.8 GB non sono utilizzabili con i kernel precedenti alla versione 2.3.21. Appresso si riportano i dettagli. Ipotizziamo l'acquisto di un nuovo disco IBM-DPTA-373420 con una capacità di 66835440 settori (34.2 GB). I kernel ante 2.3.21 riporterebbero una dimensione di 769*16*63 = 775152 settori (0.4 GB) che è una cosa spiacevole. Se fornissimo i parametri hdc=4160,255,63 da linea di comando non otterremmo alcun effetto - tale valori sarebbero ignorati. Ma cosa succede? La funzione idedisk_setup() utilizza la geometria fornita dal disco (che è 16383/16/63) e sovrascrive i valori specificati dall'utente da linea di comando che vengono utilizzati solo per impostare la geometria da passare al BIOS. La funzione current_capacity() o idedisk_capacity() ricalcola il numero dei cilindri come 66835440/(16*63)=66305 poichè tale valore è memorizzato in una variabile short diviene 769.Fino a quando lba_capacity_is_ok() non distrugge id->cyls ogni sua chiamata successiva riporterà il valore falso, di conseguenza la capacità del disco diviene 769*16*63. È disponibile un aggiornamento per diverse versioni del kernel. Una "patch" per la versione 2.0.38 può esere trovata presso: *ftp.kernel.org* . Una "patch" per la versione 2.2.12 può essere trovata presso: *www.uwsg.indiana.edu*

(potrebbero essere necessarie delle modifiche per eliminare i delimitatori del linguaggio html). I kernel della serie 2.2.14 gestiscono tali dischi. La serie 2.3.* gestisce questi dischi a partire dalla versione 2.3.21. È possibile "risolvere" il problema a livello hardware utilizzando un 11.3 (ponticello) (jumper) per impostare la dimensione di 33.8 GB. In molti casi sarà necessario un aggiornamento del 4.2 (BIOS) se si vuole utilizzare tali dischi come dischi d'avvio del sistema.

# 13 Partizioni estese e logiche

6 (In precedenza), abbiamo visto la struttura dell'MBR (settore 0): il codice del boot loader seguito da 4 voci della tavola delle partizioni da 16 byte ciascuna e dalla firma AA55. Le voci della tavola delle partizioni di tipo 5 o F o 85 (Hex) hanno un significato speciale: descrivono le partizioni *estese*, che sono porzioni di disco che verranno ripartizionate in partizioni *logiche* (una partizione estesa non è che un contenitore di partizioni logiche e non può essere utilizzata di per se stessa). Solo la posizione del primo settore di una partizione estesa è importante. Questo primo settore contiene una tavola delle partizioni con quattro voci: una per la partizione logica, una per quella estesa e due inutilizzate. In questo modo si ottiene una catena di tavole delle partizioni, sparse per il disco, dove la prima descrive tre partizioni primarie e una estesa mentre le tavole seguenti descrivono una partizione logica e la posizione della prossima tavola.

È importante capire questo: quando qualcuno fa qualcosa di stupido partizionando un disco si chiede: "I miei dati ci sono ancora?" La risposta è: sì. Ma, se si creano partizioni logiche verrà riscritta la tavola delle partizioni che le descrive e ogni dato presente è perso.

Il programma sfdisk mostrerà l'intera catena. Ecco un esempio:

```
# sfdisk -l -x /dev/hda

Disk /dev/hda: 16 heads, 63 sectors, 33483 cylinders
Units = cylinders of 516096 bytes, blocks of 1024 bytes, counting from 0
```

```
   Device Boot Start      End   #cyls   #blocks   Id  System
/dev/hda1          0+     101     102-    51376+  83  Linux
/dev/hda2        102     2133    2032   1024128   83  Linux
/dev/hda3       2134    33482   31349  15799896    5  Extended
/dev/hda4          0        -       0         0    0  Empty

/dev/hda5       2134+    6197    4064-  2048224+  83  Linux
    -           6198    10261    4064   2048256    5  Extended
    -           2134     2133       0         0    0  Empty
    -           2134     2133       0         0    0  Empty

/dev/hda6       6198+   10261    4064-  2048224+  83  Linux
    -          10262    16357    6096   3072384    5  Extended
    -           6198     6197       0         0    0  Empty
    -           6198     6197       0         0    0  Empty
...
/dev/hda10     30581+   33482    2902-  1462576+  83  Linux
    -          30581    30580       0         0    0  Empty
    -          30581    30580       0         0    0  Empty
    -          30581    30580       0         0    0  Empty

#
```

È possibile costruire una tavola delle partizioni scorretta. Molti kernel entrano in un loop se qualche partizione estesa punta a se stessa o ad una precedente nella catena. È possibile avere due partizioni estese in una tabella delle partizioni così che la tabella si divida (questo capita per esempio con fdisk che non riconosce ogni 5, F e 85 come partizioni estese e che crea una 5 dopo una F). Programmi non standard tipo fdisk possono risolvere queste situazioni anche se richiedono del lavoro manuale per riparare le partizioni. Il kernel di Linux accetta una divisione nei livelli più esterni. In questo modo è possibile avere due catene di partizioni logiche. Talvolta può essere utile - per esempio, si può usare il tipo 5 che è visto dal DOS e il tipo 85, invisibile al DOS, così che il DOS FDISK non vada in crash perché la partizione logica supera il cilindro 1024. Di solito si ha bisogno di `sfdisk` per eseguire queste operazioni.

# 14 Risoluzione dei problemi

Molte persone pensano di avere problemi anche quando non ce ne sono. Oppure pensano che i loro problemi siano legati alla geometria del disco quando alla fine dei conti la geometria non c'entra nulla con il problema. Tutto quello che ho esposto potrà sembrare complicato ma maneggiare la geometria dei dischi è estremamente facile: non fate niente e tutto funzionerà correttamente; o forse passate a LILO l'opzione 'linear' se al boot non andate oltre a LI. Guardate i messaggi del kernel al boot e ricordate: meno giocherellate con le geometrie (specificando il numero di testine e cilindri a LILO o a fdisk o dalla riga di comando al kernel) e più possibilità avete che tutto funzioni. Per dirla in poche parole, va già tutto bene così com'è configurato in maniera predefinita.

E ricordate che Linux non utilizza in nessuna riga di codice la geometria del disco per questo motivo non avrete problemi utilizzandolo dovuti alla geometria. La geometria del disco è utilizzata solamente da LILO e da fdisk. Di converso, se LILO fallisce l'inizializzazione del sistema ci può essere un problema legato alla geometria. Se sistemi operativi diversi non capiscono la tavola delle partizioni allora c'è un problema legato alla geometria. Non c'è nient'altro. In particolare se il montaggio delle periferiche ha dei problemi, state tranquilli non sono legati alla geometria del disco ma a qualcos'altro.

## 14.1 Problema: avviando il sistema da un disco SCSI viene assegnata ai dischi IDE una geometria errata.

È possibile che un disco fornisca la sua geometria in modo errato. Il kernel di Linux interroga il BIOS per conoscere la geometria di hd0 e hd1 (i driver del BIOS indicati come 80H e 81H) e assume che questi dati siano per hda e hdb. Ma in un sistema SCSI, in cui i primi due dischi possono essere SCSI, può succedere che al quinto disco, che è il primo disco IDE hda, sia assegnata la geometria di sda. Questo tipo di problemi si risolvono facilmente fornendo i parametri della geometria 'hda=C,H,S' in fase di inizializzazione o in /etc/lilo.conf.

## 14.2 Un non problema: dischi identici possono avere geometria diversa?

'Posseggo due dischi eguali da 10 GB della IBM. Tuttavia fdisk fornisce delle dimensioni diverse. Guarda:

```
# fdisk /dev/hdb
Disk /dev/hdb: 255 heads, 63 sectors, 1232 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot  Start      End   Blocks   Id  System
/dev/hdb1           1     1232  9896008+  83  Linux native
# fdisk /dev/hdd
Disk /dev/hdd: 16 heads, 63 sectors, 19650 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot  Start      End   Blocks   Id  System
/dev/hdd1           1    19650  9903568+  83  Linux native
```

Cos'è capitato?'

Cosa sta succedendo? Bene, prima di tutto questi dischi sono entrambi da 10 giga: le dimensioni di hdb sono 255*63*1232*512 = 10133544960, e quelle di hdd 16*63*19650*512 = 10141286400, così non c'è nulla di errato e il kernel li vede entrambi come dischi da 10.1 GB. Perché la differenza di dimensione? È dovuta al fatto che il kernel acquisisce i dati per i primi due dischi IDE da BIOS e il BIOS ha rimappato hdb in modo che abbia 255 testine (e 16*19650/255=1232 cilindri). L'arrotondamento ti costa almeno 8 MB.

Se vuoi rimappare hdd devi passare, come fatto prima, i parametri 'hdd=1232,255,63' al kernel in fase di inizializzazione.

## 14.3 Un non problema: fdisk vede molti blocchi in meno di df?

fdisk ti dice quanti blocchi ci sono nel disco. Se tu crei un filesystem sul disco, diciamo con mke2fs, allora questo filesystem ha bisogno di spazio per mantenere le informazioni relative a se stesso - di solito circa il 4% della dimensione del filesystem, di più se il numero degli inode è alto. Per esempio:

```
# sfdisk -s /dev/hda9
4095976
# mke2fs -i 1024 /dev/hda9
mke2fs 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
...
204798 blocks (5.00%) reserved for the super user
...
# mount /dev/hda9 /somewhere
# df /somewhere
```

```
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/hda9            3574475      13  3369664      0%   /mnt
# df -i /somewhere
Filesystem           Inodes   IUsed   IFree  %IUsed Mounted on
/dev/hda9            4096000      11 4095989     0%  /mnt
#
```

Abbiamo una partizione con 4095976 blocchi, creiamo un filesystem di tipo ext2 nella stessa, la montiamo e scopriamo che ha solo 3574475 blocchi - 521501 blocchi (12%) sono "persi" perché dedicati agli inode e ad altre informazioni. È da osservare che la differenza tra il numero di blocchi totale 3574475 e il numero di blocchi disponibili per gli utenti 3369664 è pari a 13 blocchi in più dei 204798 riservati al root. Quest'ultimo numero può esserre modificato utilizzando tune2fs. Questa opzione '-i 1024' è ragionevole solo per sistemi che devono contenere molti piccoli file come le news e simili. Il valore predefinito sarebbe:

```
# mke2fs /dev/hda9
# mount /dev/hda9 /somewhere
# df /somewhere
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/hda9            3958475      13  3753664      0%   /mnt
# df -i /somewhere
Filesystem           Inodes   IUsed   IFree  %IUsed Mounted on
/dev/hda9            1024000      11 1023989     0%  /mnt
#
```

Ora solo 137501 blocchi (3.3%) sono usati per gli inode, così abbiamo guadagnato rispetto a prima 384 MB (ogni inode accupa 128 byte). D'altro canto, questo filesystem può contenere al massimo 1024000 di file (più che sufficienti), contro i 4096000 (troppi) di prima.