HDs grandes: mini-COMO

Andries Brouwer, `aeb@cwi.nl`
Traducción: Fco. J. Montilla, `pacopepe@insflug.org` v1.0, 960626

Todo lo referente a geometrías de disco duro, así como el límite de las BIOS/SO relacionados con particiones más allá de los 1024 cilindros.

Índice General

1 El Problema

Suponga que tiene un disco duro con más de 1024 cilindros. Suponga que encima emplea un sistema operativo que hace uso de la *BIOS*. Si es así, entonces tiene un problema, ya que el habitual interfaz de la BIOS a operaciones de E/S con discos, la INT13, usa un campo de 10 bits para el cilindro en el que se produce una operación de E/S, por lo que cilindros más allá del 1024 resultarán inaccesibles.

Afortunadamente, Linux no hace uso de la BIOS, por lo que no resulta un problema.

Bueno, a excepción de dos aspectos:

1. **Cuando arranca el sistema**, Linux no está ejecutándose todavía, y por tanto, no puede ahorrarle los problemas relacionados con la BIOS. Esto acarrea ciertas consecuencias para con LILO y gestores de arranque similares.

2. **Es necesario, para todos los sistemas operativos** que compartan un disco el coincidir en dónde están las particiones. En otras palabras, si usa Linux, y por ejemplo, *Dos* en un disco, ambos deberán

> interpretar la tabla de particiones del mismo modo. Esto resulta en determinadas consecuencias con el núcleo de Linux y `fdisk`.

Lo que sigue es una descripción bastante detallada de todos los detalles relevantes. Téngase en cuenta que empleé el `kernel` 2.0.8 como fuente de referencias; para otras versiones la cosa puede variar ligeramente.

2 Arrancando

Cuando el sistema arranca, la BIOS lee el sector 0 (conocido como *MBR*, *Master Boot Record*, o *Registro de Arranque Principal*) del primer disco (o de un floppy), y *salta* al código allí residente –algún tipo cargador *bootstrap*[1], generalmente–.

Los pequeños programas de *bootstrap* que allí se encuentran no poseen controladores de disco propios, típicamente, y emplean servicios de la BIOS para ello. Esto significa que el núcleo de Linux sólo puede arrancar cuando reside enteramente antes de los primeros 1024 cilindros.

Este problema se puede resolver fácilmente, asegurándose de que el núcleo (y quizás otros ficheros empleados durante la fase de arranque, como los ficheros de *mapeo* de LILO) residen en una partición que se encuentra en su totalidad en los primeros 1024 cilindros de un disco al que la BIOS puede acceder –esto significa que probablemente sea el primer o segundo disco–.

Otro punto a considerar es que tanto el *cargador* de arranque y la BIOS deben asentir en cuanto a la geometría del disco. Podría ser útil dar a `LILO` la opción '`linear`'. Más sobre esto a continuación.

3 Geometría de los discos y particiones.

Si tiene varios sistemas operativos en sus discos, cada uno puede estar utilizando una o varias particiones. El más mínimo desacuerdo en cuanto a dónde están dichas particiones puede acarrear catastróficas consecuencias.

El MBR contiene una *tabla de particiones* describiendo donde están las particiones (primarias). Hay 4 entradas en dicha tabla, para 4 particiones primarias[2], y cada una tiene el siguiente aspecto:

```
struct partition {
        char active;    /* 0x80: arrancable, 0: no arrancable */
        char begin[3];  /* CHS para el primer sector */
        char type;
        char end[3];    /* CHS para el ultimo sector */
        int start;      /* numero de sector en 32 bit (contado desde 0) */
        int length;     /* numero de sectores 32 en bit */
};
```

(donde CHS se refiere a Cilindros/Cabezas/Sectores).

Por tanto, la información es redundante: la localización de la partición se da tanto por los campos de 24 bits `begin` y `end`, como por los campos de 32 bits `start` y `length`.

Linux sólo usa los campos `start` y `length`, y puede por tanto manejar particiones no mayores de 2ˆ32 sectores, o lo que es lo mismo, particiones de más de 2 TeraBytes. Lo cual es doscientas veces el tamaño de los discos disponibles actualmente, por lo que será suficiente para los próximos 10 años o así.

[1] N del T: El término *bootstrap* procede de la expresión anglosajona "*To pull oneself up by one's bootstraps* ", que viene a significar "*rehacerse por sí sólo, por sus propios medios, o por méritos propios, etc*"

[2] N del T: o para 3 primarias y 1 extendida

Desafortunadamente, la llamada a la INT13 de la BIOS emplea los CHS codificados en tres bytes, con 10 bits para el número de cilindro, 8 para el número de cabeza, y 6 para el numero de sector de pista.

Los números posibles de cilindro son 0-1023, de cabeza 0-255, y de sector de pista 1-63 (sí, los sectores de una pista se cuentan desde 1, no desde 0). Con esos 24 bits se puede acceder a 8455716864 bytes (7.875 GB), doscientas veces más de las capacidades de disco disponibles en 1983.

Todavía más desafortunadamente, los interfaces IDE estándar permiten 256 sectores/pista, 65536 cilindros y 16 cabezas. Esto por sí mismo permite el acceso a 2^37 = 137438953472 bytes (128 GB), pero combinado con la restricción de la BIOS a 63 sectores y 1024 cilindros hace que sólo queden 528482304 bytes (504 MB) accesibles.

Esto no es suficiente para los discos de hoy en día, por lo que la gente recurre a todo tipo de triquiñuelas, tanto vía *hardware* como *software*.

4 Traslaciones y Managers de disco

A nadie le importa cuál es la geometría 'real' de un disco. De hecho, incluso el número de sectores por pista es variable –hay más sectores por pista conforme nos acercamos al borde exterior del disco– por lo que no hay un número de sectores por pista 'verdadero'.

Para el usuario resulta más conveniente considerar un disco como una simple serie[3] lineal de sectores numerados como 0, 1, ..., y dejar a la controladora el trabajo de encontrar en qué lugar del disco reside un sector dado.

Esta numeración lineal se conoce como **LBA**. Las direcciones lineales correspondientes a (c,h,s)[4]para un disco con geometría (C,H,S)[5] corresponden a c*H*S + h*S + (s-1). Todas las controladoras SCSI "hablan" LBA, junto con algunas IDE.

Si la BIOS convierte los (c,h,s) de 24 bits a LBA y le pasa eso a una controladora que "entienda" LBA, entonces vuelven a ser accesibles 7.875 GB. No es suficiente para todos los discos, pero no deja de ser constituir una mejora.

Téngase en cuenta que que aquí CHS, tal y como los usa la BIOS, no tiene nada que ver con la 'realidad'.

Algo similar funciona cuando la controladora no "habla" LBA, pero la BIOS sí sabe hacer la traslación. (En el *setup* esto se suele indicar como "*large*".) La BIOS presentará una geometría (C',H',S') al sistema operativo, usando (C,H,S) para hablar con la controladora del disco. Normalmente $S = S', C' = C/N$ y $H' = H * N$, donde N es la menor potencia de dos que asegura que C' <= 1024 (a fin de que se malgaste la menor capacidad posible al redondear en C' = C/N). Nuevamente, esto nos proporciona acceso a 7.875 GB.

Si la BIOS no sabe nada acerca de 'LBA' o 'Large', podemos recurrir a ciertas soluciones *software*. Los *Disk Managers* o *Gestores de Disco* como *OnTrack* o *EZ-Drive* reemplazan las rutinas de gestión de disco por otras suyas. Esto se lleva a cabo normalmente haciendo residir el código del gestor de disco en el MBR y sectores subsecuentes (OnTrack llama a este código *DDO*: *Dynamic Drive Overlay*[6], de modo que sea arrancado antes que cualquier otro sistema operativo. Esa es la razón por la que se pueden tener problemas arrancando desde disquete cuando se ha instalado un Gestor de Disco.[7]

El efecto es más o menos el mismo que cuando se hacen traslaciones vía BIOS, pero –especialmente cuando haya distintos sistemas operativos en el mismo disco– con la salvedad de que pueden causar bastantes problemas.

[3]N del T: "*array*" en el original.
[4]N del T: c=cilindros h=cabezas s=sectores; parámetros lógicos.
[5]N del T: parámetros físicos
[6]N del T: algo así como *Superposición Dinámica de Disco*.
[7]N. del T. Esto no significa que no pueda hacerse de modo seguro; los gestores suelen proveer utilidades para hacer disquetes especiales de arranque

Linux soporta *OnTrack Disk Manager* desde la versión de núcleo 1.3.14, y *EZ-Drive* desde la versión 1.3.29. Más detalles a continuación.

5 Traslación de disco al núcleo en discos IDE.

Si el núcleo de Linux detecta la presencia de algún gestor de disco en un disco IDE, tratará de remapear el disco del mismo modo en que lo haya hecho el gestor de disco, de modo que Linux 'vea' el mismo particionamiento del mismo que bajo DOS con OnTrack o EZ-Drive.

No obstante, **NO** se produce remapeo alguno cuando se especifica la geometría en la línea de comandos[8], por lo que una opción como '`hd=cyls,heads,secs`' puede acabar perfectamente con la compatibilidad de un gestor de disco.

El remapeo se lleva a cabo probando con 4, 8, 16, 32, 64, 128, 255 Cabezas (manteniendo H*C constante) hasta que o bien C <= 1024 o H = 255.

Los detalles vienen a continuación. –los títulos de las subsecciones son las que aparecen en los mensajes de arranque correspondientes–. Tanto aquí como en cualquier otra parte de este texto, los tipos de partición se darán en notación hexadecimal.

5.1 EZD

EZ-Drive se detecta por el hecho de que la primera partición primaria es de tipo 55. La geometría se remapea como se describió anteriormente, desechando la tabla de particiones del sector 0 –en lugar de ello, se lee del sector 1–. Los números de bloque del disco no sufren cambio alguno, tan sólo se redirigen las operaciones de escritura al sector 0 al sector 1. Se puede cambiar este comportamiento recompilando el kernel con la definición

```
#define FAKE_FDISK_FOR_EZDRIVE  0
```

en `ide.c`.

5.2 DM6:DDO

OnTrack DiskManager (en el primer disco) se detecta por el hecho de que la primera partición primaria es de tipo 54. La geometría se remapea como se describió anteriormente, y el disco se desplaza enteramente 63 sectores (por lo que el antiguo sector 63 pasa a ser el sector 0). Posteriormente, un MBR nuevo (con su correspondiente tabla de particiones) es leído del nuevo sector 0. Por supuesto este desplazamiento se lleva a cabo para hacer sitio al *DDO* –a esto se debe a que no se produzca dicho desplazamiento en los demás discos–.

5.3 DM6:AUX

OnTrack DiskManager (en otros discos) es detectado por la circunstancia de que la primera partición primaria es de tipo 51 o 53. La geometría se remapea como describimos antes.

5.4 DM6:MBR

Una versión más antigua de OnTrack DiskManager se detecta no por el tipo de partición, sino por la firma. (Se comprueba que el *offset* encontrado en los bytes 2 y 3 del MBR no es superior a 430, la abreviatura

[8] N del T: Paso de parámetros a LILO o loadlin al arrancar Linux

encontrada en este *offset* es igual a `0x55AA`, y está seguido por un byte impar.). Nuevamente, la geometría se remapea como anteriormente.

5.5 PTBL

Para finalizar, hay un test que intenta deducir si existe traslación a partir de los valores `start` y `end`[9] de las particiones primarias: Si alguna partición posee un cilindro de comienzo y fin menor de 256, sectores de comienzo y fin 1 y 63 respectivamente, y como cabezas finales 31, 63 o 127, entonces dado que acabar las particiones en el límite de un cilindro es algo "no estandarizado" y dado que además los interfaces IDE usan como mucho 16 cabezas, se deduce que está activa alguna traslación de BIOS, y la geometría se remapea para usar 32, 64 o 128 cabezas respectivamente. (Puede que haya una inconsistencia aquí, y `genhd.c`, ¿no debería haber comprobado dos bits mayores por orden del número de cilindro?). No obstante, no se lleva a cabo remapeo alguno cuando la noción que se tiene en este momento de la geometría ya es de 63 sectores por pista con al menos tantas otras cabezas (dado que esto significaría que el remapeo ya se ha producido).

6 Consecuencias

¿Qué significa todo esto? Para los usuarios de Linux tan sólo una cosa: deben de asegurarse de que `LILO` y `fdisk` usan la geometría correcta, donde 'correcta' se define para `fdisk` como la misma geometría usada por los otros sistemas operativos presentes en el mismo disco, y para `LILO` como la geometría que hará posible una interacción exitosa con la BIOS en el momento del arranque. (Normalmente ambos coinciden).

¿Cómo averigua `fdisk` la geometría? Pregunta al núcleo, empleando la llamada `ioctl HDIO_GETGEO`. No obstante, el usuario puede interponerse a dicha geometría, interactivamente o en la línea de comandos.

¿Cómo averigua `LILO` la geometría? Pregunta al núcleo, usando la llamada `ioctl HDIO_GETGEO`. No obstante, el usuario puede descartar dicha geometría empleando la opción '`disk=`'. Se puede pasar también la opción `linear` a LILO, que almacenará en tal caso direcciones LBA en lugar de CHS en su fichero de mapeo, averiguando la geometría a usar al arrancar (mediante el empleo de de la `INT13`, función `8` para preguntar la geometría del disco).

Cómo sabe el núcleo qué responder? Veamos, para comenzar, el usuario puede haber especificado una geometría explícitamente con la opción, en la línea de comandos[10] '`hd=cyls,heads,secs`'. En cualquier caso distinto, el núcleo preguntará al *hardware*.

6.1 Detalles IDE

Permítame elaborar. El controlador[11] IDE tiene cuatro fuentes de información acerca de la geometría. La primera, (*G_user*) es la especificada por el usuario en la línea de comandos. La segunda (*G_bios*) es la Tabla de Parámetros de Disco Duro de la BIOS. (para el primer y segundo disco solamente). Esto se lee al arrancar el sistema, antes de cambiar a modo 32 bits. La tercera (*G_phys*) y cuarta (*G_log*) son proporcionadas por la controladora IDE como respuesta al comando *IDENTIFY* –son las geometrías "físicas" y "lógicas actuales"–.

Por otra parte, el controlador (*driver*, software;) precisa dos valores para la geometría: por una parte `G_fdisk`, devuelto por la llamada `ioctl HDIO_GETGEO`, y por otra, `G_used` que es empleada actualmente para las operaciones de E/S. Tanto `G_fdisk` como `G_used` son inicializadas a `G_used` si se especifica éste, a `G_bios` cuando dicha información está presente de acuerdo a la *CMOS*, y a `G_phys` en los demás casos. Si `G_log` parece razonable, entonces `G_used` se inicializa como él. En cualquier otro caso, si `G_used` no parece

[9] N del T: `comienzo` y `final` respectivamente

[10] N del T: Al *prompt* de `LILO`, el "`LILO boot:`" que aparece al encender el ordenador si `LILO` está instalado.

[11] N del T: "*driver*" en el original, software

razonable y sí lo parece `G_phys`, entonces `G_used` se inicializa a `G_phys`. 'Razonable' aquí significa que el número de cabezas esté en el rango 1-16.

Dicho con otras palabras: la línea de comandos descarta la BIOS, y determinará lo que `fdisk` va a ver, pero si especifica una geometría que ya ha sufrido traslación, (más de 16 cabezas), para operaciones de E/S a nivel núcleo será sustituida por lo retornado por el comando *IDENTIFY*.

6.2 Detalles SCSI

La situación para los discos SCSI es ligeramente diferente, ya que los comandos SCSI usan ya números de bloque lógicos, por lo que la 'geometría' es completamente irrelevante para las operaciones de E/S.

No obstante, el formato de la tabla particiones continúa siendo el mismo, por lo que `fdisk` tendrá que inventarse alguna geometría, y también usará `HDIO_GETGEO` aquí –de hecho, `fdisk` no distingue entre discos IDE o SCSI–. Como uno puede ver a raíz de la descripción detallada anterior, los distintos *drivers* inventan cada uno, una geometría diferente de algún modo. Un gran *follón*, de hecho.

Si no usa DOS o similar, evite todas las configuraciones con traslación extendida, empleando simplemente 64 cabezas, y 32 sectores por pista (para un bonito y práctico 1 MB por cilindro), si es posible, de modo que no aparezcan problemas cuando cambie el disco de una controladora a otra.

Algunos controladores de discos SCSI (`aha152x, pas16, ppa, qlogicfas, qlogicisp`) son tan paranoicos con la compatibilidad con DOS que no permitirán a un sistema sólo-Linux emplear más de 8Gb. Esto es un fallo.

¿Qué es "geometría real"? La respuesta más sencilla es que no existe tal cosa. Y si la hubiese, no debería querer saberla, y desde luego NUNCA, JAMÁS decírsela a `LILO` o `fdisk`.

Esto es un asunto a tratar exclusivamente entre la controladora SCSI y el disco. Permítame repetírselo: sólo los tontos le dicen a `fdisk/LILO`/kernel la verdadera geometría de un disco SCSI.

Si aún así es usted curioso e insiste, debería preguntarle al propio disco. Existe el importante comando *READ CAPACITY* que proporcionará el tamaño total del disco, así como existe el comando *MODE SENSE*, que proporciona el número de cilindros y cabezas (información que no puede ser cambiada) de la *Página de Geometrías de Disco Duro* (página 04), y que extrae de la *Página de Formateo* (página 03) el número de de bytes por sector, así como de sectores por pista. Este último número es típicamente dependiente de las marcas[12], variando el número de sectores por pista –las pistas externas tienen más sectores que las internas.

El programa Linux `scsiinfo` proporcionará esta información. Existen más detalles y complicaciones, pero está claro que nadie (probablemente ni siquiera el sistema operativo) quiere usar esta información.

Más aún, en lo que a nosotros concierne respecto a `fdisk` y `LILO`, obtendremos respuestas típicamente como C/H/S=4476/27/171 –valores que no pueden ser empleados por `fdisk` porque la tabla de particiones reserva sólo 10/8/6 bits para C/H/S.

¿Entonces de dónde averigua la llamada al kernel `HDIO_GETGEO` su información? O bien de la controladora SCSI, o bien mediante cultas averiguaciones. Algunos controladores parecen pensar que queremos saber la 'realidad', pero por supuesto lo único que queremos saber es qué utilizarán los FDISK de DOS u OS/2 (o el AFDISK de Adaptec, etc).

Nótese que el `fdisk` de Linux necesita los números H y S de las cabezas y sectores por pista para convertir de números de sectores LBA a localizaciones c/h/s, pero el número C de cilindros no tienen nada que ver en esta conversión. Algunos controladores usan (C,H,S) = (1023,255,63) para indicar que la capacidad del disco es al menos 1023*255*63 sectores. Esto no resulta muy afortunado, ya que no revela la capacidad actual, y limitará a los usuarios de la mayoría de las versiones de `fdisk` a alrededor de 8 Gb máximo en sus discos –una verdadera limitación hoy en día–.

[12] N del T: No marca comercial, marca física.

En la descripción que sigue, M denota la capacidad total del disco, y C, H, S el número de cilindros, cabezas y sectores por pista. Basta con proporcionar H, S si tenemos en cuenta a C como definido por M / (H*S).

Por defecto, H=64, S=32.

aha1740, dtc, g_NCR5380, t128, wd7000:

H=64, S=32.

aha152x, pas16, ppa, qlogicfas, qlogicisp:

H=64, S=32 a menos que C > 1024, en cuyo caso H=255, S=63, C = min(1023, M/(H*S)). (C por tanto es truncado, y H*S*C no es una aproximación a la capacidad del disco M. Esto causará confusión en la mayoría de las versiones de fdisk.) El código *ppa.c* emplea M+1 en lugar de M y dice que ello se debe a un error en *sd.c*, en el que a M le falta 1.

advansys:

H=64, S=32 a menos que C > 1024 y más aún si la opción '> 1 GB' está activado en la BIOS, en cuyo caso H=255, S=63.

aha1542:

Pregunte a la controladora cuál de los dos esquemas de traslación posibles está en uso, y emplee tanto H=255, S=63 como H=64, S=32. En el último caso habrá un mensaje al arrancar: "`aha1542.c: Using extended bios translation`".

aic7xxx:

H=64, S=32 a menos que C > 1024, y además o bien el parámetro de arranque "`extended`" haya sido especificado en el arranque, o si el bit "`extended`" ha sido especificado en la SEEPROM o BIOS, en cuyo caso H=255, S=63.

buslogic:

H=64, S=32 a menos que C >= 1024, y que además se haya configurado la controladora para hacer traslaciones extendidas, en cuyo caso si M < 2^22 entonces H=128, S=32; de otro modo, H=255, S=63. No obstante, tras hacer esta elección para (C,H,S), se lee la tabla de particiones, y si para alguna de las tres posibilidades (H,S) = (64,32), (128,32), (255,63) el valor endH=H-1 aparece por alguna parte, entonces es usado dicho par (H,S), y un mensaje será mostrado al arranque: "`Adopting Geometry from Partition Table`".

fdomain:

Averigüe la información acerca de la geometría de la Tabla de Parámetros de Disco de la BIOS, o lea la tabla de particiones y use H=endH+1, S=endS para la primera partición, teniendo en cuenta que no esté vacía, o use H=64, S=32 para M < 2^21 (1 GB), H=128, S=63 para M < 63*2^17 (3.9 GB) y S=63 en otro caso.

in2000:

Emplee los primeros (H,S) = (64,32), (64,63), (128,63), (255,63) que hagan que C <= 1024. En el último caso, trunque C a 1023. Use los primeros de (H,S) = (64,32), (64,63), (128,63), (255,63)

seagate:

Lea C,H,S del disco. (¡Horror!) Si C o S es demasiado grande, ponga S=17, H=2 y vaya doblando H hasta que C <= 1024. Esto significa que H será establecida a 0 si M > 128*1024*17 (1.1 GB). Esto es un error de programación (*bug*).

ultrastor y u14_34f:

Uno de los tres mapeos ((H,S) = (16,63), (64,32), (64,63)) es empleado dependiendo del modo de mapeo de la controladora.

Si el driver no especifica la geometría, volveremos a realizar una averiguación inteligente usando la tabla de particiones, o usando la capacidad total del disco.

Mire la tabla de particiones. Dado que por convención las particiones terminan en el límite de un cilindro, podemos, dado un `end = (endC,endH,endS)` de una partición, poner simplemente H = `endH+1` y S = `endS`. (Recuerde que los sectores son numerados a partir de 1). De un modo más preciso, se hace lo siguiente: Si hay alguna partición que no esté vacía, escoja la partición con el `beginC` mayor. Para dicha partición, mire a `end+1`, calculados ambos añadiendo `start` y `length` y asumiendo que estas particiones terminan en los límites de un cilindro. Si ambos valores concuerdan, o si `endC` = 1023 y `start+length` es múltiplo integral de `(endH+1)*endS`, asuma entonces que dicha partición está realmente alineada con el límite de un cilindro, y ponga H = `endH+1` y S = `endS`.

Si esto falla, bien debido a que no hay particiones, o porque poseen tamaños extraños, tenga en cuenta entonces únicamente la capacidad del disco M. Algoritmo: ponga H = M/(62*1024) (redondeando hacia arriba), S = M/(1024*H) (redondeando hacia arriba), C = M/(H*S) (redondeando hacia abajo).

Esto tiene el efecto de producir un (C,H,S) con C siendo como mucho 1024 y S como mucho 62.

7 Anexo: El INSFLUG

El *INSFLUG* forma parte del grupo internacional *Linux Documentation Project*, encargándose de las traducciones al castellano de los Howtos (Comos), así como la producción de documentos originales en aquellos casos en los que no existe análogo en inglés.

En el **INSFLUG** se orienta preferentemente a la traducción de documentos breves, como los *COMOs* y *PUFs* (**P**reguntas de **U**so **F**recuente, las *FAQs*. `:)`), etc.

Diríjase a la sede del INSFLUG para más información al respecto.

En la sede del INSFLUG encontrará siempre las **últimas** versiones de las traducciones: `www.insflug.org`. Asegúrese de comprobar cuál es la última versión disponible en el Insflug antes de bajar un documento de un servidor réplica.

Se proporciona también una lista de los servidores réplica (*mirror*) del Insflug más cercanos a Vd., e información relativa a otros recursos en castellano.

Francisco José Montilla, `pacopepe@insflug.org`.