# Administración de Discos

## Tabla de contenidos

Geometría de los Discos Duros.
Límites a la Geometría de los Discos IDE
Problemas Causados por Límites a la Geometría de los Discos IDE.
Particiones del Disco.
La consola de administración de discos
Configuración de la Consola
Discos básicos y dinámicos.
Creación de Particiones.
Creación de Volúmenes.
Utilidades
Diskpart
Sistemas de Ficheros
Copias de Seguridad
Carpetas y ficheros
Estado del Sistema

### Geometría de los Discos Duros.

Los discos duros son el medio de almacenamiento masivo y permanente por excelencia en los ordenadores. Existen dos grandes grupos de discos en función de su interfaz con el ordenador, IDE y SCSI. En esencia, ambos grupos son equivalentes, salvo en aspectos de rendimiento, fiabilidad y precio. Difieren, eso sí, en laslimitaciones que el software de sistemas ha impuesto de formaartificial a los discos IDE en el mundo de los PCs.Un disco almacena su información en uno o más platos, disponiendo de una cabeza lectora para cada una de las dos caras del plato (aunque en algunas ocasiones una cara no es utilizada). Cada cara está dividida en varios anillos concéntricos, denominados pistas. Esta división es debida a que el plato gira sobre su eje, y la cabeza lectora se desplaza longitudinalmente hacia o desde el eje. A su vez, cada pista está subdividida en sectores, todos ellos de igual capacidad, 512 bytes en la gran mayoría de los casos. Todos los platos de un disco están unidos y también lo están entre si las cabezas lectoras. El conjunto de pistas que se encuentran bajo todas las cabezas lectoras recibe el nombre de cilindro.

Resumiendo, la capacidad de un disco puede describirse indicando su número de cilindros, cabezales y sectores por pista. Por ejemplo, un disco con 4096 cilindros, 16 cabezales y 63 sectores por pista alberga un total de:

4096 x 16 x 63 = 4.128.768 sectores de 512 bytes

2.113.929.216 bytes

2.064.384 Kbytes

2.016 Mbytes =

1'96875 Gbytes.

Hay que tener en cuenta que el fabricante del disco dirá que su disco tiene 2^30 Gbytes. Los fabricantes de discos asumen que un Gbyte equivale a mil millones de bytes, no a $2^{30}$ bytes. Por cierto, los nuevos estándares dan la razón a los fabricantes, y nos indican que deberíamos utilizar los nuevos términos Kibytes, Mibytes, Gibytes para expresar las correspondientes potencias de 2. Sin embargo, en este documento no vamos a utilizar este estándar.

### Límites a la Geometría de los Discos IDE

Existen diferentes límites a la geometría descrita que han sido impuestos (artificialmente) por el hardware o el software. Los tres más importantes se explican a continuación, seguidos de un apartado que comenta cómo se han superado hasta la fecha.

### La especificación ATA

Según la especificación establecida por los fabricantes de discos duros, la forma de indicar al controlador del disco qué sector deseamos acceder es mediante su número de cilindro, de cabezal y de sector. La especificación también establece un número máximo de bits para cada valor, lo cual condiciona el número más grande de cilindros, cabezales y sectores que

podemos direccionar. Este máximo número de bits es, para cada valor, el siguiente:

- Para el número de cilindro : 16 bits.
- Para el número de cabezal : 4 bits.
- Para el número de sector: 8 bits.

Por tanto, según la especificación ATA, un disco duro puede direccionar, como mucho:

65536 cilindros * 16 cabezales * 256 sectores por pista.

Si multiplicamos este número máximo de sectores por 512 bytes, el resultado es de 127'5 Gbytes, es decir, el tamaño teórico máximo de disco duro de la especificación ATA.

En el momento de escribir estas líneas, este límite está prácticamente a punto de alcanzarse, dado que ya se venden discos IDE de 100Gb de capacidad.

**Las Rutinas de Disco Clásicas de la BIOS.**

Algunos sistemas operativos antiguos para PC, como por ejemplo MSDOS, utilizan la BIOS como la forma natural de acceder a los discos, entre otros dispositivos. Es el mismo caso que muchos de los cargadores de los sistemas operativos actuales. En estos casos, el sistema operativo informa a la BIOS de qué número de sector desea acceder, y la BIOS traduce esta petición al controlador del disco, según la especificación ATA presentada anteriormente.

En este caso, la propia BIOS presenta al sistema operativo una interfaz de funciones en la que tiene también un número de bits reservados para direccionar el cilindro, cabezal y sector. La cantidad máxima de bits que las funciones ``clásicas'' de acceso reservaban para cada valor son las siguientes:

- Para el número de cilindro : 10 bits.
- Para el número de cabezal : 8 bits.
- Para el número de sector: 6 bits.

Por tanto, utilizando las rutinas tradicionales de la BIOS, un disco duro puede direccionar, como mucho:

1024 cilindros * 256 cabezales * 63 sectores por pista

El hecho de que el número de sectores por pista sea de 63 en vez de 64 proviene del hecho de que, según el mecanismo tradicional de direccionamiento de los discos (denominado CHS\footnote{Su nombre viene del acrónimo inglés *Cylinder-Head-Sector*, los sectores se numeran desde 1 en vez de numerarlos desde 0.

Multiplicando esta cantidad de sectores por 512 bytes, el valor resultante es de 7'84 Gbytes, que es el tamaño máximo de disco duro que reconocen esas BIOS tradicionales. Incluso en BIOS más modernas, que incorporan otras funciones de acceso, esa limitación sigue existiendo para el software que utiliza aquellas funciones clásicas para acceder al disco. Este es el caso, por ejemplo, del ancestral sistema DOS y, hasta hace poco tiempo, también de LILO, el cargador por excelencia del sistema operativo Linux.

**La Limitación Conjunta de las dos Anteriores.**

En un primer momento, la limitación real no fue la anterior, sino la limitación combinada de las dos anteriores. Es decir, puesto que los discos duros no pueden tener más de 16 cabezales (porque así lo han decidido los fabricantes), la limitación de la BIOS aún se restringía más, dando como resultado los siguientes números:

- Para el número de cilindro : 10 bits.
- Para el número de cabezal : 4 bits.

- Para el número de sector: 6 bits.

Es decir:

1024 cilindros * 16 cabezales * 63 sectores por pista.

Multiplicando por 512 bytes, esta cantidad de sectores permitía un total de \B{504 Mbytes}, una cantidad supuestamente inalcanzable en los años de los primeros sistemas para PC, pero que apenas 15 años después resultó claramente insuficiente.

**Superando las Limitaciones.**

La aparición de discos de más de 504 Mb, con más de 1024 cilindros, causó serios problemas, ya que millones de ordenadores con DOS instalado no podían utilizarlos directamente.

La forma de salvar esta primera limitación consistió en aprovechar que la BIOS permitía un número mayor de cabezales que la especificación ATA, aunque menos cilindros. Por tanto, el objetivo consistía en implementar a nivel de BIOS una traducción que, manteniendo inalterable el número máximo de sectores del disco, ofreciera al sistema operativo un número superior (ficticio) de cabezales (hasta 256) y un número proporcionalmente inferior de cilindros (hasta 1024). Internamente la BIOS realiza la traducción de ese número de sector virtual al número de sector real.

En este sentido, el modo de direccionamiento LBA (o Logical Block Addressing) resulta particularmente interesante. En este modo de direccionamiento, el controlador del disco ofrece a los niveles superiores la visión de que el disco está formado por un vector lineal de sectores. Por tanto, los tres valores que identifican un sector en el disco (cilindro, cabezal y sector) se especifican como un único número entre 0 y el número máximo de sector. A nivel de la especificación ATA, el número de sector sería un número de 28 bits (16 + 4 + 8). Al aparecer discos que permitían direccionamiento

LBA, las BIOS fueron modificadas para utilizar también este modo y sus 24 bits de direccionamiento (10 + 8 + 6) se utilizaron para especificar un número lineal entre 0 y 2^24, que luego la BIOS pasa directamente al controlador del disco, permitiendo llegar naturalmente hasta el máximo de 7'84 Gb.

Finalmente, esa barrera de los 7'84 Gb se ha superado incluyendo nuevas funciones en las BIOS, con una especificación LBA que utiliza un número mayor de bits. De esta forma se puede llegar hasta el máximo de la especificación ATA.

**Problemas Causados por Límites a la Geometría de los Discos IDE.**

Estas limitaciones en los discos típicos para PC han ocasionado limitaciones en algunos sistemas operativos. A continuación se revisan los sistemas más populares:

1. DOS, Windows 3.x, Windows 95, Windows NT 3.x. DOS utiliza las rutinas clásicas de la BIOS para acceder al disco. Por tanto, DOS no puede utilizar más de 1024 cilindros. El modo LBA es imprescindible y el tamaño máximo de disco soportado, tal como se ha explicado anteriormente, es de 7'84 Gbytes. Los sistemas Windows que aparecieron a continuación heredaron esta restricción de DOS por motivos de compatibilidad.
2. Windows 95 OSR2, Windows 98, Windows ME. No tienen ningún problema. No utilizan las rutinas clásicas de la BIOS y por tanto pueden utilizar más de 1024 cilindros.
3. Windows NT 4.0. Este sistema tiene soporte para discos grandes desde su aparición. No obstante, el proceso de instalación de NT está basado en DOS, con lo cual, la partición donde reside NT debe estar antes de la barrera de los 1024 cilindros.

En resumen, en nuestros días apenas nos estamos librando de las

restricciones que impuso el diseño original de la BIOS. Sólo en las últimas versiones de los sistemas operativos para PC, esas limitaciones no suponen un problema de instalación o de uso.

**Particiones del Disco.**

En el primer sector de un disco duro reside el denominado MBR (o Master Boot Record). En estos 512 bytes residen el código inicial de carga del sistema operativo, la tabla de particiones primarias y la firma del disco.

El código de carga más frecuente es el que define el sistema operativo DOS, el cual se encarga de buscar la partición de arranque, cargar en memoria el primer sector de dicha partición y cederle el control. Éste es el método utilizado por todos los sistemas operativos de Microsoft. Este código de carga puede recuperarse (reinstalarse) con el mandato DOS FDISK /MBR, el cual deja intacta la tabla de particiones.

Otros códigos de carga bastante utilizados son los cargadores que vienen con el sistema operativo Linux: LILO y GRUB. En realidad, el código que se ubica en el MBR es el correspondiente a la primera fase del proceso de arranque. Estos cargadores son más complejos y flexibles que el código de carga de DOS, y utilizan la información que reside en el directorio /boot de Linux para determinar qué sistema operativo debe cargarse.

El código de carga se utiliza tan sólo en el disco principal del sistema (es decir, el disco maestro del interfaz IDE primario). Hay BIOS que permiten arrancar de otros discos duros, pero generalmente aparecen numerosos problemas al utilizar esta opción.

La tabla de particiones está formada por cuatro entradas, donde cada una de ellas describe una potencial partición primaria. Los detalles de cada entrada son algo oscuros, y su utilización varía sensiblemente de un sistema operativo a otro. No obstante, simplificando un poco, cada entrada indica, para la partición que describe, la siguiente información:

1. El sector del disco donde comienza.
2. Su tamaño.
3. Si es una partición de arranque (activa).
4. Su tipo.

DOS, Windows 3.x y Windows 95 representan el inicio de la partición utilizando la tripleta <cilindro,cabezal,sector>, con lo cual el tamaño máximo de una partición es de 7'84 Gbytes (debido a la limitación de los 1024 cilindros). El resto de sistemas representan el inicio de la partición indicando cuál es el sector lógico donde comienza (viendo al disco como un vector de sectores lógicos). Estos sistemas utilizan una palabra de 32 bits, lo que permite 2^32 sectores, equivalente a 2 Tbytes (2 x 1024 Gbytes). En este caso, aún estamos algo lejos de disponer de discos de estas características.

El indicador de partición de arranque es utilizado tan sólo por el código de carga de DOS. Linux ignora por completo esta información.

Existen numerosos tipos de particiones, en función de su utilización o de su organización interna (establecida por el sistema operativo que la defina). Existe una convención que establece el identificador de cada tipo de partición como un número concreto en hexadecimal. La siguiente abla expone los tipos de particiones más habituales.

| Tipo | Uso | Limitación |
|---|---|---|
| 0 | Partición vacía | |
| 5 | Partición extendida | 1024 cilindros (7'84GB) |
| 6 | DOS FAT 16 | 2 GB |
| 7 | OS2 o NTFS | 2TB |

| | | |
|---|---|---|
| | | |
| b | Windows 95 FAT 32 | 2TB |
| f | Extendida Windows 95 | 2TB |
| 80 | Old Minix | - |
| 82 | Linux Swap | - |
| 83 | Linux Native | 2TB |

Una partición extendida es un contenedor para otras particiones, a las cuales se les denomina particiones lógicas. Sólo una de las particiones primarias puede declararse como extendida. La representación de las unidades lógicas se realiza utilizando una lista enlazada que reside dentro de la partición extendida, por lo que no hay límite en cuanto al número de particiones lógicas que se pueden crear. La partición extendida convencional ha tenido que ser cambiada por la nueva partición extendida Windows 95, ya que la primera no puede abarcar discos mayores de 7'84 Gbytes.

### Instalando E2fsprogs-1.27

Estimación del tiempo de construcción: 0.80 SBU
Estimación del espacio de disco requerido: 13 MB

**Instalación de E2fsprogs**

Instala E2fsprogs ejecutando los siguientes comandos:

```
mkdir ../e2fsprogs-build &&
cd ../e2fsprogs-build &&
```

```
../e2fsprogs-1.27/configure --prefix=/usr --with-root-prefix="" \
   --enable-elf-shlibs &&
make &&
make install &&
make install-libs &&
install-info /usr/share/info/libext2fs.info /usr/share/info/dir
```

### Explicación de los comandos

**--with-root-prefix="":** La razón por la que proporcionamos esta opción es por la configuración del fichero Makefile de e2fsprogs. Algunos programas son esenciales para el uso del sistema cuando, por ejemplo, /usr todavía no ha sido montada (como el programa e2fsck). Por lo tanto, estos programas y librerías corresponden a los directorios /lib y /sbin. Si no se pasase esta opción al comando ./configure de e2fsprogs, colocaría estos programas en /usr, que no es lo que queremos.

**--enable-elf-shlibs:** Esto crea librerías compartidas que algunos programas de este paquete pueden usar.

**make install-libs:** Esto instala las librerías compartidas que han sido construidas.

### Contenido de E2fsprogs

Última versión comprobada: 1.27.

### Programas

badblocks, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, fsck, fsck.ext2, fsck.ext3, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs y uuidgen

**Descripciones**

**badblocks**

badblocks se usa para buscar bloques dañados en un dispositivo (normalmente una partición de disco).

**chattr**

chattr cambia los atributos de un fichero en un sistema de ficheros ext2 de Linux.

**compile_et**

compile_et es usado para convertir una tabla con códigos de error y sus mensajes asociados en un fichero fuente C apropiado para usar con la librería com_err.

**debugfs**

El programa debugfs es un depurador de sistemas de ficheros. Puede usarse para examinar y cambiar el estado de un sistema de ficheros ext2.

**dumpe2fs**

dumpe2fs muestra la información del superbloque y de los grupos de bloques del sistema de ficheros presente en un determinado dispositivo.

**e2fsck y fsck.ext2**

e2fsck y fsck.ext2 se usan para chequear, y opcionalmente reparar, sistemas de ficheros ext2.

**e2image**

e2image se usa para salvar información crítica de un sistema de ficheros ext2

en un fichero.

**e2label**

e2label muestra o cambia la etiqueta de un sistema de ficheros ext2 situado en el dispositivo especificado.

**fsck**

fsck se usa para chequear, y opcionalmente reparar, un sistema de ficheros Linux.

**fsck.ext3**

fsck.ext3 se usa para chequear, y opcionalmente reparar, un sistema de ficheros ext3.

**lsattr**

lsattr muestra los atributos de un fichero en un sistema de ficheros ext2.

**mk_cmds**

La utilidad mk_cmds toma como entrada un fichero de tabla de comandos y genera como salida un fichero fuente C preparado para usarlo con la librería del subsistema, libss.

**mke2fs and mkfs.ext2**

mke2fs se usa para crear sistemas de ficheros ext2 en un dispositivo (normalmente una partición de disco). mkfs.ext2 hace lo mismo que mke2fs.

**mkfs.ext3**

mkfs.ext3 se usa para crear un sistema de ficheros ext3.

**mklost+found**

mklost+found se usa para crear un directorio lost+found en el directorio de trabajo actual de un sistema de ficheros ext2. mklost+found reserva una serie de bloques de disco en el directorio para que sean usados por e2fsck.

**resize2fs**

resize2fs se usa para redimensionar sistemas de ficheros ext2.

**tune2fs**

tune2fs ajusta los parámetros de un sistema de ficheros ext2.

**uuidgen**

El programa uuidgen crea un nuevo identificador universal único (UUID) usando la librería libuuid. El nuevo UUID puede considerarse razonablemente único por muchos UUID que se hayan creado en el sistema local o en otros sistemas en el pasado o en el futuro.

*Imagen de cfdisk*