HOWTO - Disques de grande capacité

Andries Brouwer, aeb@cwi.nl,
version française par Xavier Serpaggi, xavier.serpaggi@libertysurf.fr v2.2z, 2 février 2002

Tout sur la géométrie des disques durs et la limite des 1024 cylindres.

Pour obtenir une version toujours à jour, mais en anglais, de ce document reportez-vous à la page *www.win.tue.nl* .

1 Énoncé du problème

Supposons que vous ayez un disque dur de plus de 1024 cylindres. Supposons également que vous ayez un système d'exploitation qui utilise l'ancienne interface INT13, d'Éntrée/Sortie sur disques. Dans ce cas, vous avez un problème, parce que cette interface utilise un champ de 10 bits pour coder les cylindres sur lesquels sont effectuées les Éntrées/Sorties, de telle manière que les cylindres 1024 et au-delà sont inaccessibles.

Heureusement, Linux ne se sert pas du BIOS, donc il n'y a pas de problème.

Sauf peut-être pour deux choses :

(1) Quand vous démarrez votre système, Linux ne fonctionne pas encore et ne peut donc pas vous préserver des problèmes liés au BIOS. Cela a certaines conséquences pour LILO et d'autres programmes d'amorçage du même acabit.

(2) Il est nécessaire que tous les systèmes d'exploitation qui se partagent un même disque dur se mettent d'accord sur la position physique de chaque partition. En d'autres termes, si vous utilisez Linux et disons, DOS sur un seul disque dur, alors les deux se doivent d'interpréter la table des partitions de la même manière. Cela a quelques conséquences pour le noyau de Linux et pour `fdisk`.

Vous trouverez ci-dessous une description assez poussée de tous les détails importants. Prenez en compte le fait que j'utilise comme référence un noyau dans sa version 2.0.8. D'autres versions pourront donc présenter quelques différences.

2 Résumé

Vous avez un nouveau disque de grande capacité. Que faire ? Bon, du côté logiciel il faut utiliser `fdisk` (ou mieux, `cfdisk`), pour créer les partitions, ensuite `mke2fs` pour créer un système de fichiers et enfin `mount` pour faire le lien entre ce nouveau système de fichiers et l'arborescence déjà existante.

Il n'est pas nécessaire de lire ce HOWTO à partir du moment où, de nos jours, il n'y a *pas* de problème avec les disques de grande capacité. La grande majorité des problèmes constatés est due au fait que les gens pensent qu'il peut y avoir un problème et installent un gestionnaire de disques durs, ou passent en mode expert dans `fdisk`, ou encore spécifient explicitement une géométrie de disque à LILO ou sur la ligne de commande du noyau.

Cependant, les domaines dans lesquels interviennent typiquement les problèmes sont :

- un matériel ancestral ;
- plusieurs systèmes d'exploitation sur le même disque dur ;
- le démarrage.

Conseil :

Pour les disques durs SCSI de grande capacité : Linux les a très tôt supportés. Il n'y a rien à faire.

Pour les disques durs IDE de grande capacité (au-delà de 8,4 Go) : procurez-vous un noyau stable récent (2.0.34 ou plus). Normalement, tout doit se passer correctement, surtout si vous avez eu la sagesse de ne pas dire au BIOS de faire des conversions du type LBA ou assimilé.

Pour des disques durs IDE de capacité vraiment importante (au-delà de 33,8 Go) : reportez-vous à la section 12.1 (Problèmes de l'IDE avec des disques durs de 34 Go et plus) plus bas dans ce document.

Si LILO reste bloqué au démarrage, il faut essayer de spécifier `5.1 (linear)` dans le fichier de configuration `/etc/lilo.conf` (et si `linear` était déjà présent, essayez sans). Si vous avez une version récente de LILO (21.4 ou mieux), le mot-clé `lba32` devrait vous permettre de démarrer de n'importe où sur le disque. Cela signifie en fait que la limite des 1024 cylindres a disparue (bien entendu, LILO est un peu fragile et il peut être plus pratique d'utiliser un gestionnaire de démarrage différent).

Il y a des problèmes de géométrie qui peuvent être résolus en passant explicitement une géométrie au noyau/LILO/fdisk.

Si vous avez une vieille version de `fdisk` et qu'il vous met des messages d'erreur du type 6 ('overlapping partitions') : ignorez-les ou vérifiez en utilisant `cfdisk` que tout va effectivement bien.

Pour le HPT366, reportez-vous au *Linux HPT366 HOWTO* .

Si, au moment du démarrage, le noyau ne peut pas lire la table des partitions, envisagez la possibilité que UDMA66 ait-été sélectionné alors que le contrôleur, le câble ou bien le disque dur, ne supportent pas ce mode. Dans ce cas, quoi que vous fassiez, vos tentatives de lecture resteront vaines et, tenter de lire la table des partitions est la première chose que fait le noyau. Assurez-vous que UDMA66 n'est pas utilisé.

Si vous pensez que quelque chose cloche dans la taille de votre disque dur, assurez-vous que vous n'êtes pas en train de confondre 3 (unités) binaires et décimales et sachez que l'espace libre rapporté par `df` pour un disque vide, est inférieur de quelques centièmes à la taille de la partition, ce à cause d'un en-tête de gestion.

Si le noyau rapporte deux tailles différentes pour un média amovible, cela veut dire que l'une est donnée par le média lui-même et l'autre par le disque/la disquette. Cette seconde valeur est égale à zéro dans le cas où aucun disque/disquette n'est présent.

Maintenant, si vous pensez qu'il y a tout de même des problèmes, ou simplement si vous êtes curieux, lisez la suite.

3 Unités et tailles

Un kilo-octet (Ko) est égal à 1000 octets (NdT : un octet se dit byte en anglais et est abrégé avec un 'B' en majuscule. À ne pas confondre avec un bit, qui se dit bit et qui est abrégé avec un 'b' en minuscule !). Un Méga-octet (Mo) est égal à 1000 Ko. Un Giga-octet (Go) est égal à 1000 Mo. Un Téra-octet (To) est égal à 1000 Go. Ceci est la *norme dans le Système International* (SI).

Cependant, il y a des personnes qui utilisent la conversion 1 Mo=1024000 octets et parlent de disquettes de 1,44 Mo et des personnes qui pensent que 1 Mo=1048576 octets. Là, je me reporte au *nouveau standard* et j'écris Ki, Mi, Gi, Ti pour les unités binaires, de telle sorte que les disquettes ont une taille de 1440 Kio (1,47 Mo, 1,41 Mio), 1 Mio est égal à 1048576 octets (1,05 Mo), 1 Gio représente 1073741824 octets (1,07 Go) et 1 Tio vaut 1099511627776 octets (1,1 To).

D'une manière assez normale, les constructeurs de disques durs suivent la norme SI et utilisent des unités décimales. Cependant, les messages de démarrage du noyau Linux (pour les noyau qui ne sont pas très récents) et quelques programmes de type `fdisk` utilisent les symboles MB et GB (Mo et Go en français) pour les unités binaires, ou binaires-décimales mélangées. Donc, avant que vous ne pensiez que votre disque est plus petit que ce qu'on vous avait promis lors de son achat, calculez sa vraie taille en unités décimales (ou simplement en octets).

En ce qui concerne la terminologie et les abréviations des unités binaires, *Knuth* propose une *alternative* qui est d'utiliser KKo, MMo, GGo, TTo, PPo, EEo, ZZo, YYo et de les dénommer *grand kilo octet*, *grand méga octet*, ... *grand yota octet*. Il écrit : *"Remarquez que le fait de doubler la lettre a une connotation à la fois binaire et d'amplitude."* C'est une bonne proposition – *grand giga octet* sonne mieux que *gibi octet*. Cependant, pour le sujet qui est le nôtre, la seule chose importante est de mettre l'accent sur le fait qu'un méga octet contient précisément 1000000 octets et qu'il est nécessaire d'employer d'autres termes ou d'autres abrévations si vous voulez désigner autre chose.

3.1 Taille d'un secteur

Dans le cadre de ce texte, un secteur a une taille de 512 octets. Cela est pratiquement toujours vrai, mais certains disques Magnéto-Optiques par exemple, utilisent une taille de secteur égale à 2048 octets et toutes les capacités données ci-dessous doivent être multipliées par quatre. (Si vous utilisez `fdisk` sur de tels disques, assurez-vous d'avoir une version 2.9i ou supérieure et passez-lui l'option `-b 2048`.)

3.2 Taille d'un disque

Un disque avec C cylindres, H têtes (NdT : tête se dit *head* en anglais, d'où l'abréviation !) et S secteurs par piste possède en tout C×H×S secteurs et peut stocker C×H×S×512 octets. Par exemple, si sur un disque dur il est écrit C/H/S=4092/16/63, alors celui-ci a 4092×16×63=4124736 secteurs et peut contenir 4124736×512=2111864832 octets (2,11 Go). Il y a une convention dans l'industrie qui consiste à donner C/H/S=16383/16/63 pour les disques durs de plus de 8,4 Go et donc la taille du disque ne peut plus être déduite des valeurs C/H/S rapportées par ce dernier.

4 Accès à un disque dur

Si on veut lire ou écrire quelque chose à partir de, ou sur un disque dur, il faut spécifier une position sur ce disque, en donnant par exemple un numéro de secteur ou de bloc. Si le disque dur est de type SCSI, alors ce numéro de secteur va directement au moteur de commande SCSI et est compris par le disque. Si le disque dur est de type IDE et qu'il utilise le mode LBA, alors il se passe exactement la même chose. Mais si le disque dur est vieux, RLL, MFM ou IDE avant l'apparition du LBA, alors l'électronique qui lui est attachée attend un triplet (cylindre, tête, secteur) pour désigner l'endroit voulu.

La correspondance entre la numérotation linéaire et cette notation tridimensionnelle est la suivante : pour un disque dur avec C cylindres, H têtes et S secteurs/pistes, la position (c,h,s) en 3D, ou la notation CHS, est la même que la position c×H×S+h×S+(s-1) en notation linéaire ou bien LBA. (Le -1 est dû au fait que traditionnellement les secteurs sont numérotés à partir de 1 et non 0, dans cette notation 3D.)

En conséquence, pour pouvoir utiliser un très vieux disque non-SCSI, il faut connaître sa *géométrie*, c'est-à-dire les valeurs de C, H et S. (Si vous n'avez pas d'information à ce sujet, vous pouvez toujours fouiller cette mine : *www.thetechpage.com* .)

4.1 Accès disques du BIOS et la limite des 1024 cylindres

Linux ne se sert pas du BIOS, mais d'autres systèmes d'exploitation le font. Le BIOS, qui existait avant le temps du LBA, offre avec INT13 des routines d'Éntrée/Sortie disque qui prennent (c,h,s) comme arguments.

(Plus précisément : `AH` sélectionne la fonction à exécuter, `CH` correspond aux 8 bits de poids faible du numéro de cylindre, `CL` a dans ses bits 7-6 les deux bits de poids fort de ce même numéro et dans ses bits 5-0 le numéro du secteur, `DH` est le numéro de la tête et `DL` est le numéro du lecteur (80h ou 81h). Cela explique en partie l'agencement de la table des partitions.)

Donc, nous obtenons un CHS codé sur trois octets, avec 10 bits pour le numéro du cylindre, 8 bits pour le numéro de tête et 6 bits pour le numéro de secteur sur la piste (numéroté de 1 à 63). Il s'ensuit que le numéro de cylindre peut prendre des valeurs allant de 0 à 1023 et que le BIOS ne peut pas adresser plus de 1024 cylindres.

Les logiciels DOS et Windows n'ont pas évolué quand furent introduits les disques IDE avec le support LBA et ont toujours eu besoin du soutien d'une géométrie pour gérer les disques durs, même quand cela n'a plus été nécessaire pour effectuer des Éntrées/Sorties, mais uniquement pour converser avec le BIOS. Cela signifie bien sûr que Linux a besoin du support de la géométrie du disque dur quand il est nécessaire de communiquer avec le BIOS ou avec d'autres systèmes d'exploitation, même sur des disques durs modernes.

Cet état de choses a duré pendant à peu près quatre ans. Ont alors commencé à apparaître sur le marché des disques durs qui ne pouvaient plus être adressés avec les fonctions INT13 (à cause du fait que 10+8+6=24 bits pour (c,h,s) ne peuvent pas adresser plus de 8,5 Go) et une nouvelle interface BIOS a été créée : les dénommées Fonctions INT13 Étendues, où DS:SI pointe sur un paquet représentant l'adressage du disque sur 16 octets, qui contient un nombre absolu de blocs commençant par 8 octets.

Tout doucement, le monde Microsoft semble aller vers une utilisation de ces Fonctions INT13 Étendues. Probablement, dans quelques années, plus aucun système moderne placé dans un ordinateur moderne n'aura besoin du concept de 'géométrie de disque dur'.

4.2 Histoire du BIOS et des limites de l'IDE

Spécification ATA (pour les disques durs IDE) – la limite des 137 Go

Au plus 65536 cylindres (numérotés de 0 à 65535), 16 têtes (numérotées de 0 à 15), 255 secteurs par piste (numérotés de 1 à 255), pour une capacité totale maximale de 267386880 secteurs (de 512 octets chacun), ce qui fait 136902082560 octets (137 Go). En 2001, le premier disque dur d'un capacité supérieure à cela est apparu (le Maxtor Diamondmax de 160 Go).

BIOS Int 13 – la limite des 8,5 Go

Au plus 1024 cylindres (numérotés de 0 à 1023), 256 têtes (numérotées de 0 à 255), 63 secteurs par piste (numérotés de 1 à 63) pour une capacité totale maximale de 8455716864 octets (8,5 Go). De nos jours, cela est une sérieuse limitation. Cela signifie que DOS ne peut utiliser les actuels disques de grande capacité.

La limite des 528 Mo

Si les mêmes valeurs c,h,s sont utilisées pour les appels aux fonctions Int13 du BIOS et pour les opérations d'Éntrées/Sorties du disque dur, alors les deux limitations s'ajoutent et l'on ne peut utiliser au plus que 1024 cylindres, 16 têtes, 63 secteurs par piste, ce qui donne une capacité totale maximale de 528482304 octets (528 Mo), l'abominable limite des 504 Mio pour DOS avec un vieux BIOS. Cela a commencé à poser des problèmes aux alentours de 1993 et les gens ont eu recours à toutes sortes de bidouillages, aussi bien au niveau matériel (LBA), qu'au niveau du micro-code (NdT : le 'firmware') (les conversions du BIOS), ou qu'au niveau logiciel (les gestionnaires de disques durs). Le concept de 'conversion' a été inventé (1994) : un BIOS pouvait utiliser une géométrie quand il s'adressait au lecteur et une autre géométrie, fausse celle-là, quand il parlait à DOS et faire des conversions de l'une à l'autre.

La limite des 2,1 Go (avril 1996)

Quelques vieux BIOS n'utilisent qu'un champ de 12 bits en RAM CMOS pour donner le nombre de cylindres. De ce fait, ce nombre peut être au plus égal à 4095 et l'on ne peut accéder qu'à 4095×16×63×512=2113413120 octets. Le fait d'avoir un disque dur de plus grande capacité se traduirait par un plantage au moment du démarrage. Cela a pour effet de rendre les disques avec une géométrie de 4092/16/63 assez populaires. Et encore de nos jours, beaucoup de gros disques durs ont un cavalier qui permet de faire croire qu'ils ont une géométrie de 4092/16/63. Vous pouvez également jeter un coup d'oeil à la page *plus de 2 Go* .

La limite des 3,2 Go

Il y avait un bug dans la gestion des BIOS Phoenix 4.03 et 4.04 qui bloquait le système lors de la configuration du CMOS pour des disques durs ayant une capacité supérieure à 3277 Mo. Jetez un oeil à la page *plus de 3 Go* .

La limite des 4,2 Go (février 1997)

Une conversion simple du BIOS (ECHS=CHS Étendu, parfois appelée 'Large disk support' ou simplement 'Large') fonctionne en doublant le nombre de têtes et en divisant par deux le nombre de cylindres montrés au DOS, de manière répétée, jusqu'à ce que le nombre de cylindres soit au plus égal à 1024. Maintenant, DOS et Windows 95 ne peuvent pas supporter 256 têtes de lecture et donc, dans le cas fréquent où le disque dit avoir 16 têtes, cela signifie que cette moulinette ne fonctionne que jusqu'à une taille de 8192×16×63×512=4227858432 octets (avec une fausse géométrie de 1024 cylindres, 128 têtes et 63 secteurs par piste). Remarquez que ECHS ne modifie pas le nombre de secteurs par piste, donc si ce n'est pas 63, la limite sera encore plus basse. Voyez la page *plus de 4 Go* .

La limite des 7,9 Go

Des BIOS un peu plus malins que les autres évitent le problème précédent en ajustant d'abord le nombre de têtes à 15 ('revised ECHS'), de façon qu'une fausse géométrie comportant 240 têtes soit obtenue, valable jusqu'à une taille de 1024×240×63×512=7927234560 octets.

La limite des 8,4 Go

En définitive, si le BIOS fait tout ce qu'il peut pour réussir la conversion et utilise 255 têtes et 63 secteurs par piste ('assisted LBA' ou simplement 'LBA') il peut atteindre 1024×255×63×512=8422686720 octets, soit légèrement moins que la précédente limite de 8,5 Go, cela parce que les géométries avec 256 têtes doivent être évitées. (Cette conversion utilisera pour le nombre de têtes la première valeur H, prise dans la suite 16, 32, 64, 128, 255, pour laquelle la capacité totale du disque dur tient dans 1024×H×63×512 et calcule alors le nombre de cylindres C comme étant égal à la capacité totale divisée par (H×63×512).)

La limite des 33,8 Go (août 1999)

Le prochain obstacle se présente avec une taille de 33,8 Go. Le problème est qu'avec les 16 têtes et les 63 secteurs/piste par défaut, ça donne un nombre de cylindres supérieur à 65535, qui ne rentre pas dans un short. La plupart des BIOS existants ne peuvent pas gérer de tels disques. (Jetez un oeil, par exemple à *Asus upgrades* pour une nouvelle image à flasher qui fonctionne.) Les noyaux Linux plus anciens que le 2.2.14/2.3.21 ont besoin d'un correctif. Voyez 12.1 (les problèmes posés par les disques durs IDE de 34 Go et plus) ci-dessous.

la limite des 137 Go (septembre 2001)

Comme ceci a déjà été mentionné ci-dessus, le vieux protocole ATA utilises 16+4+8=28 bits pour donner le numéro de secteur et de ce fait, ne peut pas adresser plus de 2^28 secteurs. ATA-6 décrit une extension qui permet d'adresser 2^48 secteurs, soit un million de fois plus. Les noyaux très récents incluent un support pour cette extension.

Pour une autre discussion sur ce sujet, vous pouvez consulter la page *Breaking the Barriers* et pour encore plus de détails, *IDE Hard Drive Capacity Barriers* .

Les disques durs de plus de 8,4 Go sont supposés donner leur géométrie comme étant 16383/16/63. Cela signifie en fait que la 'géométrie' est obsolète, et qu'elle ne peut plus servir à calculer la taille totale d'un disque dur.

5 Démarrage

Quand le système est mis en route, le BIOS lit le secteur 0 (connu sous le nom de MBR : le "Master Boot Record", la donnée principale d'amorçage) du premier disque dur (ou de la disquette, ou du cd-rom) et saute au code trouvé à cet endroit – en général un chargeur d'amorce. Les petits chargeurs trouvés à cet endroit n'ont, par principe, pas leur propre gestionnaire de disques durs et utilisent plutôt les services du BIOS. Cela signifie qu'un noyau Linux ne peut être chargé que s'il est entièrement situé dans les 1024 premiers cylindres du disque, à moins que vous ne possédiez un BIOS et un chargeur de d'amorce moderne : un BIOS qui supporte les fonctions INT13 Étendues et un chargeur de d'amorce qui sache les utiliser.

Ce problème (s'il existe) est résolu de manière très simple : assurez-vous que le noyau (et peut-être d'autres fichiers utilisés pendant la phase de démarrage, comme les fichiers 'map' de LILO) soit situé sur une partition qui est comprise toute entière dans les 1024 premiers cylindres d'un disque dur auquel le BIOS peut accéder – il est probable qu'il s'agisse du premier ou du second disque.

Ainsi, créez une petite partition, disons d'une taille de 10 Mo, de telle manière qu'il y ait de la place pour une poignée de noyaux, en vous assurant qu'elle soit contenue entièrement dans les 1024 premiers cylindres du premier ou du second disque dur. Montez-le en tant que répertoire `/boot` ; ainsi, LILO pourra y mettre ses propres fichiers.

La plupart des systèmes depuis 1998 utilisent un BIOS moderne.

5.1 LILO et les options 'lba32' et 'linear'

Le principal : si vous utilisez LILO comme chargeur d'amorce, assurez-vous d'avoir un LILO avec une numéro de version d'au moins 21.4 (LILO peut être trouvé à l'adresse suivante : *ftp://metalab.unc.edu/pub/Linux/system/boot/lilo/* `ftp://metalab.unc.edu/pub/Linux/system/boot/lilo/`).

Un appel à `/sbin/lilo` (le programme qui installe la carte d'amorçage) permet de stocker une liste d'adresses dans cette carte, pour que LILO (le chargeur d'amorce) sache à partir d'où lire l'image du noyau. Par défaut ces adresses sont stockée au format (c,h,s) et un appel INT13 ordinaire est utilisé au moment du démarrage.

Quand le fichier de configuration précise `lba32` ou `linear`, des adresses linéaires sont stockées. Avec l'option `lba32` les adresses linéaires sont également utilisées au moment du démarrage si le BIOS supporte les extensions INT13 étendues. Avec l'option `linear`, ou avec un vieux BIOS, ces adresses linéaires sont reconverties sous une forme (c,h,s) et au moment du démarrage des appels INT13 ordinaires sont utilisés.

De ce fait, avec l'option `lba32` il n'y a pas de problème de géométrie et il n'y a pas de limite à 1024 cylindres. Sans elle, il y a une limite à 1024 cylindres. Qu'en est-il de la géométrie ?

Le programme d'amorçage et le BIOS doivent être d'accord sur la géométrie du disque dur. `/sbin/lilo` demande la géométrie au noyau, mais il n'y a aucune garantie que la géométrie du noyau Linux coïncide avec celle qu'utilise le BIOS. Donc, la géométrie fourni par le noyau est souvent inutile. Dans de tels cas on peut faciliter la tâche à LILO en lui passant l'option `linear`. L'avantage alors est que, l'idée qu'à le noyau Linux de la géométrie, n'est plus utilisée. Le revers de la médaille est que `lilo` ne peut plus vous prévenir si une partie du noyau est stockée au-delà de la limite des 1024 cylindres et vous pouvez vous retrouver avec un système qui ne démarre plus.

5.2 Un bug de LILO

Avec les versions de LILO antérieures à la 2.1 il y a un autre désavantage : la conversion des adresses effectuée au démarrage comporte un bug. Quand c×H est supérieur ou égal à 65536, le calcul provoque un dépassement de capacité. Pour H supérieur à 64 cela donne à c une limite encore plus stricte que le bien connu c<1024 ; par exemple, avec H=255 et un vieux LILO, on doit avoir c<258 (c=cylindre où réside l'image du noyau, H=nombre de têtes du disque).

5.3 1024 cylindres ce n'est pas 1024 cylindres

Tim Williams a écrit : *"J'avais ma partition Linux en dessous des 1024 premiers cylindres et ça ne démarrait quand même pas. Au début quand je l'ai déplacée en dessous de 1 Go, les choses fonctionnaient."* Comment cela est-il possible ? En fait, c'était un disque dur SCSI avec un contrôleur AHA2940UW qui utilise soit H=64, S=32 (c'est à dire des cylindres de 1 Mio=1,05 Mo), soit H=255, S=63 (c'est à dire des cylindres de 8,2 Mo), en fonction des options du micro-code du disque dur et du BIOS. Il ne fait aucun doute que le BIOS se basait sur le premier groupe de valeurs, c'est pourquoi la limite des 1024 cylindres était trouvée à 1 Gio, alors que Linux utilisait la seconde méthode et LILO estimait que la limite était à 8,4 Go.

5.4 Plus de limite à 1024 cylindre sur une vieille machine IDE

Le chargeur d'amorce `nuni` ne fait pas appel aux services du BIOS mais accède directement aux unités IDE. Donc il est possible de le mettre sur une disquette ou sur le MBR et de démarrer de d'importe où de n'importe quel disque IDE (pas seulement les deux premiers). Vous pouvez trouver ce chargeur à *ftp://metalab.unc.edu/pub/Linux/system/boot/loaders/* `ftp://metalab.unc.edu/pub/Linux/system/boot/loaders/`

6 Géométrie du disque dur, partitions et 'overlap'

Si vous avez plusieurs systèmes d'exploitation sur vos disques durs, alors chacun utilise une ou plusieurs partitions. Un désaccord sur la localisation de ces partitions peut avoir des conséquences catastrophiques.

Le MBR contient une *table des partitions* qui décrit où se situent les partitions (primaires). Il y a 4 entrées à la table, pour 4 partitions primaires et chacune ressemble à :

```
struct partition {
        char active;    /* 0x80 : on peut demarrer avec ;
                           0    : on ne peut pas */
        char begin[3];  /* CHS pour le premier secteur */
        char type;
        char end[3];    /* CHS pour le dernier secteur */
        int start;      /* numero de secteur sur 32 bits (en commencant a 0) */
        int length;     /* nombre de secteurs sur 32 bits */
};
```

(avec CHS qui signifie Cylinder/Head/Sector – Cylindre/Tête/Secteur).

Cette information est redondante : la position de la partition est donnée à la fois par les champs `begin` et `end` codés sur 24 bits et par les champs `start` et `length` codés sur 32 bits.

Linux ne se sert que des champs `start` et `length` et ne peut de ce fait que prendre en compte des partitions qui ne comportent pas plus de 2ˆ32 secteurs, c'est-à-dire, des partitions d'au plus 2 Tio. Cette capacité est douze fois plus grande que celle des disques durs que l'on peut trouver de nos jours, alors peut-être que cela sera suffisant pour, environ, les cinq prochaines années. (Donc, les partitions peuvent être très grandes, mais il y a une sérieuse restriction avec le système de fichiers ext2 quand il est utilisé sur des machines avec des entiers codés sur 32 bits : la taille maximale d'un fichier est limitée à 2 Gio.)

DOS utilise les champs `begin` et `end` et se sert des appels INT13 du BIOS pour accéder aux disques durs ; il ne peut gérer de ce fait que des disques dont la taille ne dépasse pas 8,4 Go, même avec un BIOS qui fait des conversions. (La taille des partitions ne peut pas excéder 2,1 Go à cause des restrictions du système de fichiers FAT16.) La même chose est valable pour Windows 3.11, WfWG et Windows NT 3.*.

Windows 95 intègre la gestion des interfaces INT13 Étendues et utilise des types de partition spéciaux (c, e, f à la place de b, 6, 5) pour indiquer que l'on doit accéder à la partition de cette manière. Quand ces types de partition sont utilisés, les champs `begin` et `end` contiennent des informations factices (1023/255/63). Windows 95 OSR2 introduit le système de fichier FAT32 (types de partition b ou c), qui permet d'avoir des partitions d'au plus 2 Tio.

Qu'est-ce que c'est que ce message insensé que vous rapporte `fdisk` au sujet de partitions qui se chevauchent : 'overlapping', quand pourtant tout est en ordre ? En fait, il y a quelque chose de 'problématique' : si vous voyez les champs `begin` et `end` de telles partitions, comme DOS le fait, il y a chevauchement (et cela ne peut pas être corrigé, parce que ces champs ne peuvent pas stocker des numéros de cylindre plus grands que 1024 : il y aura toujours 'chevauchement' dès que vous aurez plus de 1024 cylindres). Cependant, si vous voyez les champs `start` et `length`, comme les voit Linux et également Windows 95 dans le cas de partitions typées c, e ou f, alors tout apparaît comme étant en ordre. Donc, ne tenez pas compte de ces avertissements quand `cfdisk` ne se plaint pas et que vous avez un disque dur avec uniquement Linux dessus. Soyez prudents quand le disque dur est partagé avec DOS. Servez-vous des commandes `cfdisk -Ps /dev/hdx` et `cfdisk -Pt /dev/hdx` pour voir la table des partitions du disque `/dev/hdx`.

6.1 Le dernier cylindre

Un nombre important de vieux IBM PS/2 utilisent des disques durs avec une carte des *défaillances* écrite à la fin du disque (le bit 0x20 dans le mot de controle de *la table de paramètrage du disque* est positionné). De ce fait, FDISK n'utilisera pas le dernier cylindre. Pour se prémunir d'éventuel problème le BIOS donne souvent un taille inférieure d'un cylindre par rapport celle réelle du disque, ce qui peut faire en tout, deux cylindres de perdus. Les disques récents ont plusieurs fonctions permettant de donner leur taille qui, en interne, s'apellent les unes les autres. Quand les deux retirent 1 pour ce cylindre réservé et quand FDISK le fait aussi, alors trois cylindres peuvent être perdus. De nos jours tout ceci n'a plus de sens mais peut fournir une explication quand on utilise différents utilitaires qui donnent des valeurs différentes pour la taille du disque dur.

6.2 Limites du cylindre

La croyance populaire racconte que les partitions doivent commencer et finir aux frontières des cylindres.

Puisque *la géométrie des disques* est quelque chose qui n'a pas de réelle existence, des systèmes d'exploitation différents inventeront des géométries différentes pour le même disque. On voit souvent un système d'exploitation utiliser une géométrie après conversion de */255/63 et un autre utiliser une géométrie avant conversion de */16/63. Du coup, il devrait être impossible d'aligner les partitions sur les limites des cylilndres, si l'on se fie à l'idée que chaque système d'exploitation a de la géométrie du disque. De plus, le fait d'activer ou non le BIOS d'une carte SCSI peut changer la fausse géométrie des disques SCSI connectés.

Par chance, pour Linux les alignements ne sont pas nécessaires (sauf pour quelques logiciels d'installation boiteux qui aiment bien être sûr que tout est d'aplomb ; ce qui peut empécher d'installer une RedHat 7.1 sur un disque aux partitions non alignées, DiskDruid n'étant pas content).

Des personnes rapportent qu'il est facile de créer des partitions non alignées sous Windows NT, sans qu'il n'y ait de problème apparent.

MSDOS 6.22 par contre, nécessite un alignement. Les secteurs sur partitions étendues qui ne sont pas sur la frontière d'un cylindre son ignorées par son FDISK. Le système lui-même se satisfait de n'importe quel alignement mais interprète les adresses de début relatives, comme si elles étaient relatives à une adresse alignée. L'adresse de départ d'une partition logique est donnée relativement, non pas à l'adresses de la partition étendue qui la décrit, mais au début du cylindre qui contient ce secteur (il n'est donc pas étonnant que, même PartitionMagic, nécessite un alignement).

Quelle est la définition de l'alignement ? FDISK de MSDOS 6.22 se comportera comme suit : 1. Si le premier secteur d'un cylindre est un secteur de table de partition, alors le reste de la piste sera inutilisé et la partition commencera à la prochaine piste. Ceci s'applique au secteur 0 (le MBR) et aux secteurs de table de partition précédant les partitions logiques. 2. Sinon la partition commence sur le premier secteur du cylindre. De plus, les partitions étendues commencent sur une frontière de cylindre. La page de manuel de `cfdisk` explique que les vieilles versions de DOS n'alignaient pas les partitions.

L'utilisation de partitions de type 85 pour les partitions étendues les rend invisible à DOS, ce qui assure que seul Linux pourra regarder ce qui s'y trouve.

Une petite aparté : sur une Sparc, la partition d'amorçage doit commencer sur une frontière de cylindre (mais rien n'est requis pour la fin).

7 Conversion et Gestionnaires de Disques Durs

La géométrie des disques durs (avec têtes, cylindres et pistes) est une notion qui date de l'âge de MFM et RLL. En ces temps là cela correspondait à une réalité physique. Aujourd'hui, avec l'IDE ou le SCSI, plus personne n'est intéressé par les 'véritables' valeurs de la géométrie d'un disque dur. En effet, le nombre de secteurs par piste est variable – il y en a plus pour les pistes proches du bord extérieur du disque – donc il n'y a pas de 'bon' nombre de secteurs par piste. Pratiquement à l'opposé : la commande IDE, INITIALIZE DRIVE PARAMETERS (91h) est utilisée pour renseigner le disque dur sur le nombre de têtes et de secteurs par piste qu'il est sensé avoir à ce moment précis. Il est assez normal de voir un gros disque dur récent qui n'a que 2 têtes, rapporter qu'il en a 15 ou 16 au BIOS, pendant que le BIOS peut à son tour dire au logiciel qui va s'en servir qu'il en a 255.

Pour l'utilisateur, le mieux est de voir le disque dur comme un tableau linéaire de secteurs numérotés 0, 1, ... et de laisser le micro-code trouver où, sur le disque dur, est situé tel ou tel secteur. Cette numérotation linéaire est appelée LBA.

Donc, à présent, la vision conceptuelle est la suivante : DOS, ou quel que soit le programme d'amorçage, converse avec le BIOS en se servant de la notation (c,h,s). Le BIOS convertit (c,h,s) en notation LBA en utilisant la géométrie factice dont l'utilisateur se sert. Si le disque dur accepte le LBA, alors cette valeur est utilisée pour les Éntrées/Sorties sur le disque. Autrement, elle est à nouveau convertie en (c',h',s') en utilisant la géométrie dont le disque se sert cette fois-là et qui est utilisée pour les Éntrées/Sorties.

Remarquez qu'il y a une légère confusion dans l'utilisation de l'expression 'LBA' : en tant que terme décrivant les possibilités d'un disque dur, cela signifie 'Linear Block Adressing' – Adressage de blocs de manière linéaire – (par opposition à un adressage CHS). En tant que terme de configuration du BIOS, il décrit la méthode de conversion parfois appelée 'Assisted LBA' – voir plus haut : 4.2 (La limite des 8,4 Go).

Un comportement à peu près identique apparaît quand le micro-code ne parle pas le LBA, mais que le BIOS connaît la conversion. (Dans la configuration il est souvent mentionné 'Large'.) Donc, le BIOS va présenter une géométrie (C,H,S) au système d'exploitation et va utiliser (C',H',S') quand il parlera au contrôleur du disque dur. Habituellement, S=S', C=C'/N et H = H'×N, où N est la plus petite puissance de deux qui garantisse C' <= 1024 (ainsi un minimum d'espace disque est perdu au moment de l'arrondi dans C'=C/N). Encore une fois, cela permet un accès à 8,4 Go maximum (7,8 Gio).

(La troisième option de configuration est 'Normal', pour laquelle aucune conversion n'est effectuée.)

Si un BIOS ne connaît pas 'Large' ou 'LBA', alors il existe quelque part une solution logicielle. Les gestionnaires de disques durs comme OnTrack ou EZ-Drive remplacent les routines de gestion de disque du BIOS par les leurs. Cela est souvent réalisé en faisant résider le code du gestionnaire de disque dans le MBR et les secteurs suivants (OnTrack nomme ce code DDO : Dynamic Drive Overlay - recouvrement dynamique de disque), comme ça, il est chargé avant n'importe quel autre système d'exploitation. C'est pourquoi on peut avoir des problèmes en démarrant depuis une disquette quand un gestionnaire de disque dur a été installé.

Le résultat est plus ou moins le même avec un BIOS qui fait des conversions - mais particulièrement, c'est quand on utilise plusieurs systèmes d'exploitation sur le même disque dur que ces gestionnaires peuvent poser beaucoup de problèmes.

Depuis sa version 1.3.14, Linux supporte le gestionnaire de disque dur OnTrack. EZ-Drive quant à lui est supporté depuis la version 1.3.29. Quelques détails supplémentaires sont donnés dans ce qui suit.

8 Conversions du noyau pour les disques durs IDE

Si le noyau de Linux détecte la présence d'un gestionnaire de disque sur un disque dur IDE, il va essayer de recartographier le disque de la même manière que l'aurait fait le gestionnaire de disque, comme ça Linux voit le même partitionnement pour, par exemple, DOS avec OnTrack ou EZ-Drive. Cependant, AUCUNE recartographie n'est effectuée quand une géométrie a été passée en ligne de commande – donc une option de la ligne de commande comme '`hd=`*cyls*`,`*têtes*`,`*secs*' peut très bien briser la compatibilité avec un gestionnaire de disque.

Si vous êtes touché par ce problème et que vous connaissez quelqu'un qui peut compiler pour vous un nouveau noyau, trouvez le fichier `linux/drivers/block/ide.c` et supprimez, dans la routine `ide_xlate_1024()`, le test `if (drive->forced_geom) { ...; return 0; }`.

La nouvelle cartographie est obtenue en essayant les valeurs 4, 8, 16, 32, 64, 128, 255 pour le nombre de têtes (H×C reste constant) jusqu'à ce que C <= 1024 ou que H=255.

Ci-dessous les détails – les titres des sous-sections sont les messages qui apparaissent dans les différents messages de démarrage. Ici et partout ailleurs dans ce texte, les types des partitions sont donnés en hexadécimal.

8.1 EZD

EZ-Drive est détecté par le fait que le type de la première partition primaire est 55. La géométrie est recartographiée comme décrit ci-dessus et la table des partitions du secteur 0 est supprimée – à la place, la table des partitions est celle lue sur le secteur 1. Le nombre de blocs du disque n'est pas changé, mais les écritures sur le secteur 0 sont redirigées vers le secteur 1. Ce comportement peut être modifié en recompilant le noyau avec `#define FAKE_FDISK_FOR_EZDRIVE 0` dans `ide.c`.

8.2 DM6 : DDO

OnTrack DiskManager (sur le premier disque dur) est détecté grâce au type 54 de la première partition primaire. La géométrie est recartographiée comme décrit ci-dessus et la totalité du disque est décalée de 63 secteurs (comme ça, l'ancien secteur 63 devient le numéro 0). Ensuite un nouveau MBR (avec une table des partitions) est lu depuis le nouveau secteur 0. Bien sûr ce décalage a pour but de libérer de la place pour le DD0 – c'est pourquoi il n'y a pas de décalage sur les autres disques durs.

8.3 DM6 : AUX

OnTrack DiskManager (sur les autres disques durs) est détecté grâce au type 51 ou 53 de la première partition primaire. La géométrie est recartographiée comme décrit ci-dessus.

8.4 DM6 : MBR

Une version plus ancienne de OnTrack DiskManager n'est pas détectée grâce au type de partition, mais par signature. (Un test est effectué pour savoir si la valeur de décalage trouvée dans les octets 2 et 3 du MBR n'est pas supérieure à 430 et si le `short` trouvé à cette valeur de décalage est égal à 0x55AA et qu'il est suivi par un octet impair.) Une fois encore, la géométrie est recartographiée comme décrit ci-dessus.

8.5 PTBL

Finalement, il y a un test qui tente de déduire une conversion à partir des valeurs `start` et `end` de la partition primaire : si n'importe quelle partition a comme secteurs de début et de fin respectivement 1 et 63 et comme dernier numéro de tête 31, 63, 127 ou 254, alors, à partir du moment où il est habituel de terminer des partitions sur une limite de secteur et qui plus est depuis que l'interface IDE utilise au plus 16 têtes, il est supposé qu'une conversion du BIOS est active et la géométrie est recartographiée pour utiliser respectivement 32, 64, 128 ou 255 têtes. Cependant, le disque n'est pas recartographié quand la vision actuelle de la géométrie a déjà 63 secteurs par piste et au moins autant de têtes (cela signifie sans doute qu'il a déjà été recartographié).

8.6 Comment se débarasser d'un gestionnaire de disque

Quand Linux détecte le gestionnaire de disque Ontrack Disk Manager il décale tous les accès disques de 63 secteurs. De la même manière, si Linux détecte EZ-Drive, tous les accès au secteur 0 seront décalés au secteur 1. Cela signifie qu'il peut s'avérer difficile de se débarasse de ces gestionnaires de disque. La plupart de ces gestionnaires ont une option de désinstallation, mais si vous avez besoin de supprimer un gestionnaire de disque, une approche peut être de donner explicitement une géométrie de disque sur la ligne de commande. De ce fait Linux saute la routine `ide_xlate_1024()` et il est du coup possible de supprimer la table des partitions ainsi que le gestionnaire de disque (rendant l'accès à toutes les données impossible) avec la commande

```
dd if=/dev/zero of=/dev/hdx bs=512 count=1
```

Les détails dépendent du numéro de version mineur du noyau. Les noyaux récents (depuis le 2.3.21) reconanissen les paramètres de démarrage tels que `hda=remap` et `hdb=noremap`. Il est alors possible de conserver ou d'éviter le décalage dû à EZD sans se soucier de ce qui est dans la table des partitions. Le paramètre de démarrage `hdX=noremap` permet également d'éviter le décalage dû au gestionnaire Ontrack Disk Manager.

9 Conséquences

Qu'est-ce que tout cela signifie ? Pour les utilisateurs de Linux seulement une chose : qu'ils doivent s'assurer que LILO et `fdisk` utilisent la bonne géométrie, où 'bonne' pour `fdisk` est définie comme la géométrie utilisée par les autres systèmes d'exploitation sur le même disque dur et pour LILO comme la géométrie qui va permettre des échanges valides avec le BIOS au moment du démarrage (en général les deux vont de pair).

Comment `fdisk` connaît-il la géométrie ? Il demande au noyau en utilisant l'ioctl `HDIO_GETGEO`. Mais l'utilisateur peut passer outre cela en précisant la géométrie de manière interactive, ou sur la ligne de commande.

Comment LILO connaît-il la géométrie ? Il demande au noyau en utilisant l'ioctl `HDIO_GETGEO`. Mais l'utilisateur peut passer outre cela en utilisant l'option '`disk=`' dans le fichier `/etc/lilo.conf` (voyez la page de manuel lilo.conf(5)). On peut également passer l'option `linear` à LILO, il va alors stocker des adresses LBA à la place des CHS dans son fichier 'map' et retrouvera la géométrie à utiliser au moment du démarrage (en utilisant la fonction 8 de INT13 pour connaître la géométrie du disque dur).

Comment le noyau sait-il répondre ? Eh bien, en tout premier lieu, l'utilisateur doit avoir spécifié de manière explicite, soit à la main soit par l'intermédiaire du chargeur d'amorce, la géométrie avec la commande en ligne du noyau '`hda=`*cyls*`,`*têtes*`,`*secs*' (voyez la page de manuel bootparam(7)). Par exemple, vous pouvez demander à LILO de fournir une telle option en ajoutant une ligne '`append="hda=`*cyls*`,`*têtes*`,`*secs*`"`' dans le fichier `/etc/lilo.conf` (voyez la page de manuel de lilo.conf(5)). Sinon, le noyau devra deviner, probablement en se servant des valeurs obtenues à partir du BIOS ou du matériel lui-même.

Il est possible (depuis Linux 2.1.79) de changer l'idée qu'a le noyau de la géométrie en utilisant le système de fichiers `/proc`. Par exemple :

```
# 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
#
```

Ceci est particulièrement utile si vous avez besoin d'un nombre tel de paramètres sur la ligne de commande, que LILO est dépassé (ce qui n'est pas difficile à accomplir).

Comment le BIOS connaît-il la géométrie ? L'utilisateur peut l'avoir donnée dans la configuration du CMOS. Peut-être aussi que la géométrie est lue depuis le disque et convertie comme précisé dans la configuration. Dans le cas de disques SCSI, où il n'y a pas de géométrie, celle que le BIOS doit inventer peut également être précisé via des cavaliers ou des paramètres de configuration (par exemple, les contrôleurs Adaptec ont la possibilité de choisir entre les valeur habituelles H=64, S=32 et les 'conversions étendues' H=255, S=63). Parfois, le BIOS lit la table des partitions pour savoir quelle était la géométrie du disque au moment du dernier partitionnement. – ceci implique l'hypothèse qu'une table des partitions valide est présente quand il y a une signature 55aa. Ceci est plutôt positif puisque il est alors possible de déplacer les disques de machine en machine. Mais le fait que le comportement du BIOS dépende du contenu du disque peut également être la source d'étranges problèmes. Par exemple, il a été *rapporté* qu'un disque de 2,5 Go était reconnu comme ayant une capacité de 528 Mo à cause du BIOS qui lisait la table des partitions et déduisait qu'il pouvait utiliser des valeurs CHS non converties. Un autre effet de ce comportement peut être lu dans ce *rapport* : des disques non partitionnés étaient plus lents que des disques partitionnés, ceci à cause du BIOS qui testait des modes 32 bits en lisant le MBR et en voyant qu'il possédait effectivement une signature 55aa.

Comment le disque connaît-il la géométrie ? En fait, le fabricant invente une géométrie qui, à un facteur près, donne la bonne capacité. De nombreux disques ont des cavaliers qui permettent de modifier la géométrie qu'ils donnent. Ceci permet d'éviter les bugs des BIOS. Par exemple, tous les disques IBM permettent à l'utilisateur de choisir entre 15 et 16 têtes et de nombreux fabricants ajoutent des cavaliers pour permettre de faire croire que le disque est plus petit que 2,1 Go ou 33,8 Go. Vous pouvez également lire la partie sur les cavaliers 11.2 (plus bas). Parfois, certains utilitaires permettent de changer le micro-code du disque dur.

9.1 Calcul des paramètres de LILO

Parfois il est utile de forcer une certaine géométrie en ajoutant 'hda=*cyls*,*têtes*,*secs*' à la ligne de commande du noyau. On voudra pratiquement toujours *secs*=63 et le but recherché en ajoutant cela est de spécifier *têtes*. (Des valeurs raisonnables de nos jours sont *têtes*=16 et *têtes*=255.) Que devra-t-on mettre pour *cyls* ? Précisément le nombre qui donnera la bonne capacité totale de C*H*S secteurs. Par exemple, pour un disque dur avec 71346240 secteurs (36529274880 octets) on calculera C comme étant 71346240/(255*63)=4441 (par exemple en utilisant le programme `bc`) et on donnera le paramètre de démarrage `hdc=4441,255,63`. Comment connaît-on la capacité totale exacte ? Par exemple,

```
# 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
```

donne deux manières de trouver le nombre total de secteurs 71346240. Les noyaux récent donnent également la taille précise dans les messages de démararge :

```
# dmesg | grep hde
hde: Maxtor 93652U8, ATA DISK drive
hde: 71346240 sectors (36529 MB) w/2048KiB Cache, CHS=70780/16/63
 hde: hde1 hde2 hde3 < hde5 > hde4
 hde2: <bsd: hde6 hde7 hde8 hde9 >
```

Les noyaux plus anciens donnent simplement les Mo et CHS. En général la valeur CHS et arrondie à l'entier inférieur, ce qui, pour la sortie ci-dessus, nous donnerait au moins 70780×16×63=71346240 secteurs. Dans cet exemple, il se trouve que ce sont les valeurs précises. La valeur en Mo peut être arrondie au lieu d'être tronquée et peut, dans les vieux noyaux, être en unités 'binaire' (Mio) plutôt que décimale. Remarquez la correspondance entre la taille en Mo donnée par le noyau et le numéro de modèle du Maxtor. Également, dans le cas des disques SCSI, le nombre précis de secteurs est donné dans les message de démarrage du noyau :

```
SCSI device sda: 17755792 512-byte hdwr sectors (9091 MB)
```

10 Détails

10.1 Détails de l'IDE - les sept géométries

Le gestionnaire IDE a cinq sources d'information concernant la géométrie. La première (G_user) est celle donnée par l'utilisateur sur la ligne de commande. La deuxième (G_bios) est la 'BIOS Fixed Disk Parameter Table' – table des paramètres de disque fixe du BIOS – (pour le premier et second disque dur seulement) qui est lue au démarrage du système, avant le passage au mode 32 bits. Les troisième (G_phys) et quatrième (G_log) sont données par le contrôleur IDE en réponse à la commande 10.1.1 (IDENTIFY) – ce sont les géométries 'physique' et 'logique du moment'.

D'un autre côté, le gestionnaire a besoin de deux valeurs pour la géométrie : d'abord G_fdisk, donnée par un ioctl `HDIO_GETGEO` et ensuite G_used, qui est effectivement utilisée pour les Éntrées/Sorties. G_fdisk et G_used sont toutes deux initialisées avec la valeur de G_user si elle est fournie, avec G_bios quand le CMOS dit que cette valeur est présente et avec G_phys autrement. Si G_log semble vraisemblable, alors G_used est positionnée à cette valeur. Sinon, si G_used n'est pas vraisemblable et que G_phys semble l'être, alors G_used prend la valeur de G_phys. Ici, 'vraisemblable' signifie que le nombre de têtes est compris dans l'intervalle 1-16.

En d'autres termes : la ligne de commande prend le pas sur le BIOS et va déterminer ce que va voir `fdisk`, mais si elle donne une géométrie convertie (avec plus de 16 têtes), alors pour les Éntrées/Sorties qu'effectuera le noyau elle sera elle-même remplacée par les valeurs que fournira la commande IDENTIFY.

Remarquez que G_bios est assez peu fiable : pour des systèmes qui démarrent depuis un périphérique SCSI, les premier et second disques durs peuvent très bien être SCSI ; et la géométrie que le BIOS aura donné pour sda sera utilisée par le noyau pour hda. Du reste, les disques durs qui ne sont pas déclarés dans la configuration du BIOS ne sont pas vus par ce dernier. Cela signifie que, par exemple, dans un système uniquement IDE où hdb n'est pas déclaré dans la configuration, les géométries rapportées par le BIOS pour les premier et second disques vont être appliquées à hda et hdc.

10.1.1 Commande IDENTIFY DRIVE

Quand on envoie la commande IDENTIF DRIVE (0xec) à un disque dur IDE, il retournera 256 mots d'information contenant de nombreux détails techniques. Concentrons nous uniquement sur ce qui joue une rôle pour la géométrie. Les mots sont numérotés de 0 à 255.

Nous trouvons ici trois informations : DefaultCHS (mots 1,3,6), CurrentCHS (mots 54-58) et LBAcapacity (mots 60-61).

Les chaînes ASCII sont composées de mots contenant chacun deux caractères, le premier étant l'octet de poids fort et le second l'octet de poids faible. Les valeurs 32-bits sont données avec le mot de poids faible d'abord. Les mots 54-58 sont positionnés à l'aide de la commande INITIALIZE DRIVE PARAMETERS (0x91). Ils n'ont un sens que quand CHS est utilisé, mais peuvent aider à trouver la taille réelle du disque dans le cas où celui-ci donne les valeurs 4092/16/63 à DefaultCHS (dans le but d'éviter les problèmes de BIOS).

Parfois, quand un cavalier force un disque à donner une fausse valeur pour LBAcapacity (le plus souvent une valeur de 66055248 secteurs pour pouvoir rester en dessous de la valeur limite de 33,8 Go), il faut disposer d'une quatrième information pour trouver la taille réelle du disque : le résultat de la commande READ NATIVE MAX ADDRESS (0xf8).

10.2 Détails pour le SCSI

La situation pour le SCSI est légèrement différente, puisque les commandes SCSI utilisent déjà les numéros de blocks logiques, donc une 'géométrie' est complètement hors de propos pour les véritables Éntrées/Sorties. Cependant, le format de la table des partitions est toujours le même, donc `fdisk` ne fait pas la différence entre des disques IDE et SCSI. Comme on peut le voir à partir de la description détaillée ci-dessus, chaque gestionnaire de disques s'invente une géométrie différente. Un gros foutoir en fait.

Si vous n'utilisez pas DOS ou un équivalent, alors évitez toute configuration qui met en jeu des conversions étendues et utilisez simplement 64 têtes, 32 secteurs par piste (cela a pour effet de donner une valeur tout à fait sympathique et commode de 1 Mio par cylindre), si possible, comme ça il n'y aura pas de problème quand vous déplacerez le disque dur d'un contrôleur à un autre. Certains gestionnaires de disque SCSI (aha152x, pas16, ppa, qlogicfas, qlogicisp) sont si sensibles au sujet de la compatibilité avec le DOS qu'ils ne permettront pas à un système uniquement Linux d'utiliser plus de 8 Gio. C'est un bug.

Quelle est la vraie géométrie ? La réponse la plus facile est qu'elle n'existe pas. Et si elle existait, vous ne voudriez pas la connaître et à coup sûr JAMAIS, AU GRAND JAMAIS ne devrez en dire quoi que ce soit à `fdisk` ou à LILO ou au noyau. C'est uniquement une histoire entre le contrôleur SCSI et le disque dur. Laissez-moi le redire : seules les personnes stupides donnent à `fdisk`, à LILO ou au noyau la véritable géométrie d'un disque SCSI.

Mais si vous êtes curieux et que vous insistez, vous devez demander au disque dur lui-même. Il y a

Exemple	Description
0 0x0040	Champ de bits : bit 6 : disque fixe, bit 7 : media amovible
1 16383	Nombre par défaut de cylindres
3 16	Nombre par défaut de têtes
6 63	Nombre par défautl de secteurs par piste
10-19 K8033FEC	Numéro de série (en ASCII)
23-26 DA620CQ0	Révision du micro-code (en ASCII)
27-46 Maxtor 54098U8	Nom du modèle (en ASCII)
49 0x2f00	Champ de bits : bit 9 : supporte le LBA
53 0x0007	Champ de bits : bit 0 : les mots 54-58 sont valides
54 16383	Nombre actuel de cylindres
55 16	Nombre actuel têtes
56 63	Nombre actuel de secteurs par piste
57-58 16514064	Nombre total actuel de secteurs
60-61 80041248	Nombre par défaut de secteurs
255 0xf9a5	*Checksum* et signature (0xa5)

l'importante commande READ CAPACITY qui donnera la capacité complète du disque dur et il y a la commande MODE SENSE, qui, dans la Rigid Disk Drive Geometry Page (page 04) donne le nombre de cylindres et de têtes (c'est une donnée qui ne peut pas être changée) et dans la Format Page (page 03) donne le nombre d'octets par secteur et de secteurs par piste. Ce dernier nombre est typiquement dépendant du rang et le nombre de secteurs par piste varie – les pistes extérieures ont plus de secteurs que les pistes intérieures. Le programme Linux `scsiinfo` donnera cette information. Il y a de nombreux détails et complications et il est clair que personne (probablement même pas le système d'exploitation) ne désire utiliser cette information. Du reste, tant que nous ne nous intéressons qu'à `fdisk` et à LILO, on a typiquement des réponses du style C/H/S=4476/27/171 – valeurs qui ne peuvent pas être utilisées par `fdisk` parce que la table des partitions ne réserve que 10,8 et 6 bits pour respectivement C/H/S.

Mais alors, d'où le `HDIO_GETGEO` du noyau tire-t-il ses informations ? Eh bien, soit du contrôleur SCSI, soit en faisant une supposition éclairée. Certains gestionnaires de disque semblent croire que nous voulons connaître la 'réalité', mais bien sûr nous ne voulons savoir que ce que FDISK de DOS ou de OS/2 (ou AFDISK de Adaptec, etc.) utiliseront.

Remarquez que le `fdisk` de linux a besoin des nombres H et S de têtes et de secteurs par piste pour convertir les nombres de secteurs LBA en adresses c/h/s, mais le nombre de cylindres C ne joue aucun rôle. Quelques gestionnaires de disque utilisent (C,H,S)=(1023,255,63) pour signaler que la capacité du disque dur est d'au moins $1023\times255\times63$ secteurs. Ce n'est pas de chance, puisqu'ils ne révèlent pas la vraie taille et limiteront les utilisateurs de la plupart des versions de `fdisk` à n'avoir accès qu'à environ 8 Gio de leur disque – une sérieuse limitation de nos jours.

Dans le texte ci-dessous, M représente la capacité totale du disque dur et C, H, S, le nombres de cylindres, de têtes et de secteurs par piste. Il suffit de donner H, S si l'on voit C comme étant défini par $M/(H\times S)$.

Par défaut, H=64 et S=32.

aha1740, dtc, g_NCR5380, t128, wd7000 :

H=64, S=32.

aha152x, pas16, ppa, qlogicfas, qlogicisp :

H=64, S=32 à moins que C ne soit supérieur à 1024, auquel cas H=255, S=63, $C=\min(1023, M/(H\times S))$. (Ainsi C est tronqué et $H\times S\times C$ n'est pas une approximation de la capacité M du disque dur. Cela va dérouter la plupart des versions de `fdisk`.) Le code de `ppa.c` utilise M+1 à la place de M et dit qu'à cause d'un bug dans `sd.c`, M est plus petit de 1.

advansys :

H=64 et S=32 à moins que C ne soit supérieur à 1024 ou que, encore mieux, l'option du BIOS '> 1 GB' ait été activée, auquel cas H=255 et S=63.

aha1542 :

Demande au contrôleur lequel des deux schémas de conversion est utilisé et se sert soit de H=255, S=63, soit de H=64, S=32. Dans le dernier cas, il y a un message au démarrage qui dit "aha1542.c: Using extended bios translation" ("aha1542.c: Utilisation du mode de conversion étendu du bios")

aic7xxx :

H=64 et S=32 à moins que C ne soit supérieur à 1024 ou que, mieux encore, le paramètre de démarrage "extended" ait été passé, ou que le bit 'extended' ait été positionné dans la SEEPROM ou dans le BIOS, auquel cas H=255, S=63. Dans Linux 2.0.36 ce mode de conversion étendu devrait toujours être automatiquement utilisé si il n'y a pas de SEEPROM, mais dans Linux 2.2.6, si le même cas se présente, le mode de conversion étendu est utilisé seulement si l'utilisateur le demande au travers du paramètre de démarrage (de ce fait, quand une SEEPROM est trouvée, le paramètre de démarrage est ignoré).

Cela signifie qu'une configuration qui fonctionne en 2.0.36 peut ne pas démarrer avec un noyau 2.2.6 (LILO nécessite alors le mot-clé `linear`, ou le noyau a besoin du paramètre `aic7xxx=extended` au démarrage).

buslogic :

H=64 et S=32 à moins que C ne soit supérieur à 1024, ou que, encore mieux, le mode de conversion étendu ait été autorisé au niveau du contrôleur, auquel cas si M<2^22 alors H=128, S=32 ; sinon H=255, S=63. Cependant, après avoir fait ce choix pour (C,H,S), la table des partitions est lue et si pour l'une des trois possibilités (H,S)=(64,32),(128,32),(255,63) la valeur endH=H-1 est vue quelque part, alors cette paire est utilisée et le message "Adopting Geometry from Partition Table" ("Adoption de la géométrie lue dans la table des partitions") est affiché au démarrage.

fdomain :

Il trouve l'information sur la géométrie dans la Table des Paramètres des Disques du BIOS (BIOS Drive Parameter Table), ou lit la table des partitions et utilise H=endH+1, S=endS pour la première partition, à condition qu'elle ne soit pas vide, ou utilise H=64, S=32 pour M<2^21 (1 Gio), H=128, S=63 pout M<63×2^17 (3.9 Gio) et H=255, S=63 dans les autres cas.

in2000 :

Il utilise le premier couple (H,S)=(64,32),(64,63),(128,63),(255,63) qui rendra C<1024. Dans le dernier cas, C est ramené à la valeur 1023.

seagate :

Il lit C,H,S depuis le disque dur. (Horreur !) Si C ou S sont trop grands, alors il fixe S=17, H=2 et double la valeur de H tant que C<=1024. Cela signifie que H sera mis à 0 si M>128×1024×17 (1.1 Gio). C'est un bug.

ultrastor et u14_34f :

Un de ces trois couples de référence est utilisé ((H,S)=(16,63),(64,32),(64,63)) en fonction du mode de cartographie du contrôleur.

Si le gestionnaire ne précise pas la géométrie, nous retombons dans un mode de supposition guidé par la table des partitions, ou par la capacité totale du disque dur.

Regardez la table des partitions. Puisque par convention les partitions se terminent à la limite d'un cylindre, nous pouvons, étant donné que `end=(endC,endH,endS)` pour n'importe quelle partition, simplement fixer H=`endH+1` et S=`endS`. (Il est rappelé que le décompte des secteurs commence à 1.) Plus précisément, voilà ce qui est fait : s'il existe une partition non vide, prendre la partition avec le plus grand `begin`. Pour cette partition, regarder `endH+1`, calculé à la fois en additionnant `start` et `length` et en supposant le fait que cette partition se termine à la limite d'un cylindre. Si les deux valeurs concordent, ou si `endC`=1023 et `start+length` est un multiple entier de `(endH+1)xendS`, alors accepter le fait que cette partition se termine effectivement sur la frontière d'un cylindre et fixer H=`endH+1` et S=`endS`. Si cela échoue, soit parce qu'il n'y a pas de partition, soit parce qu'elles ont des tailles bizarres, il faut uniquement se fier à la capacité totale M du disque dur. Algorithme : fixer H=M/(62×1024) (arrondi au chiffre supérieur), S=M/(1024×H) (arrondi au chiffre supérieur), C=M/(H×S) (arrondi au chiffre inférieur). Cela a pour effet de générer un triplet (C,H,S) avec C égal à 1024 au plus et S égal à 62 au plus.

11 Limite de Linux pour l'IDE à 8 Gio

Le gestionnaire IDE de Linux obtient la géométrie et la capacité d'un disque (et beaucoup d'autres choses) en utilisant une requête 10.1.1 (ATA IDENTIFY). Récemment encore le gestionnaire ne croyait pas en la valeur retournée pour lba_capacity si elle était plus de 10% supérieure à la capacité calculée par C×H×S. Cependant, par suite d'un accord entre les industriels, les disques IDE de grande capacité donnent les valeurs C=16383, H=16 et S=63, pour un total de 16514064 secteurs (7.8 Go) indépendamment de leur taille réelle, mais donnent cette taille réelle dans lba_capacity.

Les noyaux récents de Linux (2.0.34, 2.1.90) savent cela et agissent en conséquence. Si vous avez un noyau Linux plus ancien et que vous ne voulez pas passer à une version plus récente et que cedit noyau ne voit que 8 Gio d'un bien plus gros disque dur, alors essayez de remplacer la routine `lba_capacity_is_ok` dans `/usr/src/linux/drivers/block/ide.c` par quelque chose du genre

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

Pour une modification plus sûre, voyez le noyau 2.1.90.

11.1 Complications du BIOS

Comme nous venons de le dire, les gros disques durs donnent la géométrie C=16383, H=16, S=63 indépendamment de leur vraie taille, alors que cette dernière est visible par la valeur de LBAcapacity. Certains BIOS ne savent pas cela et convertissent ce 16383/16/63 en quelque chose qui a moins de cylindres et plus de têtes, par exemple 1024/255/63 ou 1027/255/63. Donc, le noyau ne doit pas seulement reconnaître la géométrie particulière 16383/16/63, mais également toutes ses versions mutilées par le BIOS. Depuis le noyau 2.2.2 cela est fait correctement (en prenant la vision du BIOS de H et S et en calculant `C=capacité/(H*S)`). En général, ce problème est résolu en mettant le disque en mode Normal dans la configuration du BIOS (ou, encore mieux, à None, ne le déclarant pas du tout au BIOS). Si cela est impossible parce que vous devez démarrer à partir de ce disque dur, ou l'utiliser avec DOS/Windows et que vous n'envisagez pas un passage au noyau 2.2.2 ou mieux, utilisez les paramètres de démarrage du noyau.

Si un BIOS rapporte 16320/16/63, c'est en général pour pouvoir aboutir à 1024/255/63 après conversion.

Il y a ici, un autre problème. Si le disque dur a été partitionné en utilisant une conversion de géométrie, alors le noyau peut, au moment du démarrage, voir cette géométrie qui est utilisée dans la table des partitions et rapporter `hda: [PTBL] [1027/255/63]`. Ce n'est pas bon parce que maintenant le disque dur n'est que de 8,4 Go. Cela a été corrigé dans le noyau 2.3.21. Encore un fois, les paramètres de démarrage du noyau seront utiles.

11.2 Des cavaliers pour sélectionner le nombre de têtes

Beaucoup de disques durs ont des cavaliers qui vous permettent de choisir entre des géométries à 15 ou à 16 têtes. Le réglage par défaut vous donnera une géométrie à 16 têtes. Parfois, les deux géométries adressent le même nombre de secteurs, parfois la version à 15 têtes est plus petite. Vous devez avoir une bonne raison pour changer cette valeur : Petri Kaukasoina a écrit : *"Un disque dur IBM Deskstar 16 GP de 10.1 Go (modèle IBM-DTTA-351010) avait ses cavaliers positionnés pour présenter 16 têtes par défaut mais ce vieux PC (avec un AMI BIOS) ne démarrait pas et j'ai eu à modifier les cavaliers pour le faire passer en 15 têtes. hdparm -i donne RawCHS=16383/15/63 et LBAsects=19807200. J'utilise 20960/15/63 pour avoir la capacité totale."* La géométrie 16383/15/63 n'est pas encore reconnue par le noyau, donc il est nécessaire de donner explicitement ces paramètres au démarrage. La géométrie 16383/15/63 n'est pas encore reconnue par le noyau, donc des paramètres de démarrage explicites sont ici nécessaires. Pour le positionnement des cavaliers voyez *http://www.storage.ibm.com/techsup/hddtech/hddtech.htm* .

11.3 Des cavaliers pour limiter la capacité totale

De nombreux disques durs ont des cavaliers qui vous permettent de faire apparaître leur taille inférieure à leur valeur réelle. C'est une bêtise que de le faire et probablement aucun utilisateur de Linux ne voudra utiliser cette fonctionnalité, mais certains BIOS plantent avec de gros disques. En général la solution consiste à conserver le disque entièrement en dehors du contrôle du BIOS. Cela n'est faisable que si ce disque dur n'est pas votre disque de démarrage.

11.3.1 Limitation à 2,1 Go

La première limite sérieuse fut celle des 4096 cylindres (soit, avec 16 têtes et 63 secteurs par piste, 2,11 Go). Par exemple un disque dur Fujitsu MPB3032ATU de 3,24 Go a une géométrie par défaut de 6704/15/63, mais ses cavaliers peuvent être positionnés pour qu'elle apparaisse comme étant 4092/16/63. Le disque rapportera une capacité LBA de 4124736 secteurs et de ce fait le système d'exploitation ne pourra pas deviner qu'il est en réalité plus grand. Dans un tel cas (avec un BIOS qui plante s'il sait que le disque dur est en réalité si grand et donc avec lequel les cavaliers sont nécessaires) on aura besoin de paramètres de démarrage pour donner à Linux la taille du disque.

C'est pas de chance. La plupart des disques durs ont des cavaliers qui peuvent être positionnés pour qu'ils apparaissent comme étant des 2 Go et pour qu'ils rapportent une géométrie tronquée comme 4092/16/63 ou 4096/16/63, mais qui leur permettent toujours de rapporter une capacité LBA complète. De tels disques durs fonctionneront bien et utiliseront l'intégralité de leur capacité sous Linux, que les cavaliers aient été positionnés ou pas.

11.3.2 Limitation à 33 Go

Une limite plus récente est celle des 12.1 (33,8 Go). Les noyaux Linux plus anciens que le 2.2.14/2.3.21 nécessitent l'application d'un correctif pour être capable de s'en sortir avec des disques durs IDE d'une taille plus importante que celle-là.

Avec un vieux BIOS et un disque de plus de 33,8 Go, il se peut que le BIOS se bloque et dans ce cas il devient impossible de démarrer, même lorsque le disque est retiré des options dans le CMOS. Vous pouvez regarder à cette adresse *la limite du BIOS à 33,8 Go* .

C'est pourquoi les disques de grande capacité de chez Maxtor ou IBM disposent d'un cavalier qui les font apparaitre comme ayant une taille de 33,8 Go. Par exemple, l'IBM Deskstar 37,5 Go (DPTA-353750) avec 73261440 secteurs (ce qui correspond à 72680/16/63, ou à 4560/255/63) a un cavalier qui peut être positionné de telle manière que le disque dur apparaît avec une taille de 33,8 Go et donc rapporte une géométrie de 16383/16/63 comme n'importe quel gros disque dur, mais avec une capacité LBA de 66055248 (correspondant à 65531/16/63, ou à 4111/255/63). Ceci reste valable pour les disques de grande capacité récents de chez Maxtor.

Maxtor Quand le cavalier est positionné, la géométrie (16383/16/63) et la taille (66055248) sont toutes deux conventionnelles et ne donnent aucune information sur la taille réelle. De plus, si l'on essaye d'accéder au secteur 66055248 ou plus, on obtient des erreurs d'entrée/sortie. Cependant, sur les disques Maxtor, la taille réelle peut être trouvée et mise à disposition, à l'aide des commandes READ NATIVE MAX ADDRESS et SET MAX ADDRESS. Nous pouvons présumer que c'est ce que font MaxBlast et EZ-Drive. Il existe également pour ceci un petit utilitaire Linux (*setmax.c*) ainsi qu'un correctif pour le noyau.

Il y a un détail supplémentaire à préciser en ce qui concerne les premiers gros disques de chez Maxtor : le cavalier J46 pour ces disques de 34-40 Go fait passer la géométrie de 16383/16/63 à 4092/16/63 et ne modifie pas la valeur donnée de LBAcapacity. Ceci signifie que, même si le cavalier est présent, les BIOS (les vieux Award 4.5*) se bloqueront au démarrage. Pour ce cas précis, Maxtor fournit l'utilitaire *JUMPON.EXE*

qui met à jour le micro-code pour que le cavalier J46 se comporte comme décrit ci-dessus.

Sur les disques Maxtor récents, la la commande `setmax -d 0 /dev/hdX` vous donnera également accès à la capacité maximale. Cependant, sur des disques légèrement plus anciens, un bug du micro-code ne vous permet pas d'utiliser `-d 0 setmax -d 255 /dev/hdX` vous mettra pratiquement la capacité maximale. Pour les Maxtor D540X-4K, voir plus bas.

IBM Pour IBM les choses sont pires : le cavalier limite effectivement la capacité et il n'existe pas de solution logicielle pour retrouver la vraie taille. La solution alors, n'est pas d'utiliser la cavalier mais de limiter par voie logicielle la capacité du disque avec la commande `setmax -m 66055248 /dev/hdX`. *"Comment ?"* me direz-vous – *"Je ne peux pas démarrer !"* IBM vous donne le truc : *Si un système avec un BIOS Award bloque pendant la détection des disques, redémarrez votre système et maintenez la touch F4 enfoncée pour court-circuiter l'autodétection des disques.* Si cela ne fonctionne pas, trouvez un autre ordinateur, branchez-y le disque et lancez-y la commande `setmax`. Une fois que ceci est fait, vous pouvez retourner sur votre première machine et là, la situation est identique à ce qui se passe avec un Maxtor : vous parvenez à démarrer et une fois le BIOS passé vous pouvez, soit appliquer un correctif au noyau, soit exécuter la commande `setmax -d 0` pour retrouver la capacité maximale.

Maxtor D540X-4K Les disques Maxtor Diamond Max 4K080H4, 4K060H3, 4K040H2 (alias D540X-4K) sont identiques aux disques 4D080H4, 4D060H3, 4D040H2 (alias D540X-4D), si ce n'est le configuration des cavaliers qui change. Une FAQ MAxtor donne les configurations Maître/Esclave/CableSelect pour ces disques, mais le cavalier qui limite la capacité des versions "4K" semble ne pas être documenté. Nils Ohlmeier prétent avoir trouvé de manière expérimentale que c'est le cavalier J42 (*"reserved for factory use"* – "réservé à une utilisation en usine"), à côté du connecteur d'alimentation (les disques "4D" utilisent le cavalier J46, comme les autres disques Maxtor).

Cependant, il se peut que ce cavalier non documenté se comporte comme le cavalier IBM : la machine démarre sans problème mais le disque est limité à 33 Go et `setmax -d 0` ne permet pas de retrouver la capacité maximale. Et la solution IBM fonctionne : il ne faut pas limiter la capacité du disque avec le cavalier, mais d'abord le brancher à une machine dont le BIOS est saint, le limiter par voie logicielle avec `setmax -m 66055248 /dev/hdX` et le remettre dans la première machine. Après avoir démarré, la commande `setmax -d 0 /dev/hdX` permet de retrouver la capacité maximale.

12 Limite de Linux à 65535 cylindres

L'ioctl `HDIO_GETGEO` retourne le nombre de cylindres dans un `short`. Cela signifie que si vous avez plus de 65535 cylindres, le nombre est tronqué et (pour une configuration SCSI typique avec 1 Mio de cylindres) un disque de 80 Gio peut apparaître comme ne faisant que 16 Gio. Une fois que le problème a été détecté, il est facile de l'éviter.

La convention de programmation est d'utiliser l'ioctl `BLKGETSIZE` pour obtenir la taille totale et `HDIO_GETGEO` pour connaître le nombre de têtes et de secteurs par piste et, si nécessaire, il est possible d'obtenir C avec la formule $C=taille/(H\times S)$.

12.1 Problèmes de l'IDE avec des disques durs de 34 Go et plus

Ci-dessous se trouve une discussion sur les problèmes du noyau Linux. Les problèmes liés au BIOS et au positionnement des cavalier ont-été traités 11.3.2 (ci-dessus).

Des unités d'une taille supérieure à 33,8 Go ne fonctionneront pas avec les noyaux antérieurs au 2.2.14/2.3.21. Les détails sont les suivants. Supposez que vous ayez acheté un tout nouveau disque dur IBM-DPTA-373420 qui offre une capacité de 66835440 secteurs (34,2 Go). Les noyaux plus anciens que le 2.3.21 vous diront que la taille est de 769*16*63=775152 secteurs (0,4 Go), ce qui est quelque peu étonnant. Et le fait de donner les paramètres hdc=4160,255,63 au démarrage n'aide en rien – ils sont tout simplement ignorés. Que se passe-t-il ? La routine idedisk_setup() retrouve la géométrie rapportée par le disque dur (qui est 16383/16/63) et écrase ce que l'utilisateur avait demandé sur la ligne de commande, de telle manière que les données de l'utilisateur ne sont utilisées que pour la géométrie du BIOS. La routine current_capacity() ou idedisk_capacity() recalcule le nombre de cylindres comme étant 66835440/(16*63)=66305 mais comme il est stocké dans un short, il devient 769. Comme lba_capacity_is_ok() a détruit id->cyls, tous les appels se solderont par un échec et la capacité du disque deviendra 769*16*63. Un correctif est disponible pour de nombreux noyaux. Un correctif pour le 2.0.38 peut être trouvé à *ftp.kernel.org* `ftp://ftp.us.kernel.org/pub/linux/kernel/people/aeb/` . Un correctif pour le 2.2.12 peut être trouvé à *www.uwsg.indiana.edu*

(il se peut qu'il faille le modifier un peu pour se débarrasser des tags html). Les noyaux 2.2.14 supportent ces disques durs. Dans la série 2.3.* des noyaux, le support pour ces disques existe depuis le 2.3.21. Il est également possible de résoudre ce problème avec une méthode matérielle en 11.3.2 (positionnant des cavaliers) pour limiter la taille à 33,8 Go. Dans la plupart des cas, une 4.2 (mise à jour du BIOS) sera nécessaire pour pouvoir démarrer depuis ce disque dur.

13 Partitions étendues et partitions logiques

6 (Ci-dessus), nous avons vu la structure du MBR (secteur 0) : code du programme d'amorçage suivi par 4 entrées de la table des partitions de 16 octets chacune, suivies par une signature AA55. Les entrées de la table des partitions de type 5 ou F ou 85 (en hexadécimal) ont une signification particulière : elles décrivent les partitions *étendues* : espaces qui seront ultérieurement fractionnés en partitions *logiques*. (Donc, une partition étendue n'est qu'une boîte et ne peut pas être utilisée par elle-même ; on utilise alors les partitions logiques qu'elle contient.) Ce n'est que la position du premier secteur de la partition étendue qui est important. Ce premier secteur contient une table des partitions avec quatre entrées : une est une partition logique, une est une partition étendue et deux sont inutilisées. De cette manière, on obtient une chaîne de secteurs de table des partitions, dispersés sur le disque dur, où le premier décrit trois partitions primaires et la partition étendue et chaque secteur de table des partitions qui suit décrit une partition logique et la position du prochain secteur de table des partitions.

Il est important de comprendre cela : quand les gens font des bêtises en partitionnant leur disque, ils veulent savoir : *"est-ce que mes données sont toujours là ?"* Et la réponse est généralement : *"oui"*. Mais si des partitions logiques ont été créées, alors les secteurs de table des partitions les décrivant sont écrits au début des ces partitions logiques et les données qui étaient initialement à ces emplacements sont perdues.

Le programme `sfdisk` montre la chaîne complète. Par exemple,

```
# 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

#
```

Il est possible de construire une mauvaise table des partitions. Beaucoup de noyaux entrent dans une boucle s'il y a des partitions étendues qui pointent sur elles-mêmes ou sur une partition placée avant dans la chaîne. Il est possible d'avoir deux partitions étendues dans un de ces secteurs de table des partitions, de sorte que la chaîne de la table de partitions se divise. (Cela peut arriver par exemple avec un `fdisk` qui ne reconnaît pas les partitions typées 5, F ou 85 comme étendues et qui crée une 5 à la suite d'une F.) Des programmes de type `fdisk` non standards peuvent provoquer de telles situations et quelques manipulations sont nécessaires pour les réparer. Le noyau de Linux acceptera une division au niveau le plus extérieur. Ainsi, vous pouvez avoir deux chaînes de partitions logiques. C'est parfois utile – par exemple, on peut utiliser pour l'une le type 5 afin qu'elle soit vue par DOS, et pour l'autre le type 85, invisible pour DOS, ainsi DOS FDISK ne plantera pas à cause d'une partition logique au-delà du cylindre 1024. En général il faut `sfdisk` pour créer une telle configuration.

14 Résolution des problèmes

Beaucoup de personnes pensent qu'elles ont des problèmes, alors qu'en réalité rien ne cloche. Elles peuvent également penser que leurs problèmes sont dus à la géométrie du disque, alors que cela n'a rien à voir. Tout ce que nous avons vu ci-dessus peut vous avoir paru compliqué, mais maîtriser le domaine de la géométrie des disques durs est très facile : ne faites rien du tout et tout ira bien ; ou peut-être faudra-t-il donner à LILO le mot-clé `lba32` s'il ne dépasse pas le stade LI au démarrage. Regardez bien les messages de démarrage du noyau et souvenez-vous que plus vous tripoterez les géométries (en spécifiant le nombre de têtes et de cylindres à LILO et à `fdisk` et en les passant comme argument au noyau), moins cela aura de chances de fonctionner. En gros, les valeurs par défaut sont les bonnes.

Et souvenez-vous : la géométrie des disques durs n'est utilisée nulle part dans Linux, donc les problèmes que vous pouvez avoir en vous servant de Linux ne sont pas dus à ça. En fait, la géométrie des disques durs n'est utilisée que par LILO et par `fdisk`. Donc, si LILO ne parvient pas à charger le noyau, ce peut être un problème de géométrie. Si d'autres systèmes d'exploitation ne comprennent pas la table des partitions, ce peut être un problème de géométrie. Rien d'autre. En particulier, si mount ne semble pas vouloir fonctionner, ne vous posez jamais de question sur la géométrie – le problème est ailleurs.

14.1 Problème : La géométrie de mon disque dur IDE est fausse quand je démarre depuis du SCSI.

Il est assez possible qu'un disque dur obtienne une mauvaise géométrie. Le noyau de Linux questionne le BIOS au sujet de hd0 et hd1 (les disques du BIOS numérotés 80H et 81H) et suppose que ces données sont pour hda et hdb. Mais, sur un système qui démarre depuis du SCSI, les deux premiers disques peuvent très bien être des disques durs SCSI et de ce fait il peut arriver que le cinquième disque, qui est hda, c'est-à-dire le premier disque IDE, se voie assigner une géométrie appartenant à sda. Cela est facilement résolu en donnant, au démarrage ou dans le fichier /etc/lilo.conf, les paramètres pour hda 'hda=C,H,S' avec les valeurs appropriées pour C, H et S.

14.2 Faux problème : des disques identiques ont des géométries différentes ?

"Je possède deux disques durs IBM identiques de 10 Go. Cependant, `fdisk` *donne des tailles différentes pour les deux. Voyez :*

```
# fdisk -l /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 -l /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
```

Comment cela est-il possible ?"

Que se passe-t-il ici ? Eh bien, avant tout ces disques sont réellement de 10 Giga : hdb a comme taille $255\times63\times1232\times512=10133544960$ et hdd a pour taille $16\times63\times19650\times512=10141286400$, donc tout va bien et le noyau voit les deux comme des 10 Go. Pourquoi y a-t-il cette différence de taille ? C'est parce que le noyau obtient ses données du BIOS pour les deux premiers disques IDE et le BIOS a recartographié hdb pour qu'il ait 255 têtes (et $16\times19650/255=1232$ cylindres). L'arrondi inférieur coûte ici au moins 8 Mo.

Si vous voulez recartographier hdd de la même manière, donnez au noyau l'option de démarrage 'hdd=1232,255,63'.

14.3 Faux problème : `fdisk` voit beaucoup plus d'espace que df ?

`fdisk` vous donnera le nombre de blocs qu'il y a sur le disque dur. Si vous avez créé un système de fichiers sur le disque, disons avec mke2fs, alors ce système de fichiers a besoin d'un peu de place pour sa comptabilité – typiquement quelque chose comme 4% de la taille du système de fichier, un peu plus si vous demandez beaucoup d'inodes à mke2fs. Par exemple :

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

Nous avons une partition de 4095976 blocs, créez sur cette dernière un système de fichiers ext2, montez-la quelque part et remarquez qu'elle n'a que 3574475 blocs – 521501 blocs (12%) ont été perdus en inodes et autres pour de la comptabilité. Remarquez que la différence entre le total de 3574475 blocs et les 3369664 disponibles pour l'utilisateur est égale aux 13 blocs utilisés plus les 204798 blocs réservés à root. Cette dernière valeur peut être changée à l'aide de tune2fs. Ce '-i 1024' n'est raisonnable que dans le cadre d'un spoule de forums d'utilisateurs ou quelque chose du même style, avec énormément de petits fichiers. Par défaut on mettrait :

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

À présent, seulement 137501 blocs (3,3%) sont utilisés pour les inodes, comme cela, nous disposons de 384 Mo de plus qu'avant. (Apparemment, chaque inode occupe 128 octets.) D'un autre côté, ce système de fichiers peut avoir au plus 1024000 fichiers (plus qu'assez), contre 4096000 (trop) auparavant.