Large Disk HOWTO

Andries Brouwer, aeb@cwi.nl; svensk översättning: Joel Rosdahl, joel@rosdahl.net v1.1, 18 May 1998; svensk översättning 20 augusti 1998

Allt om hårddisk-geometrier och 1024-cylindersgränsen för hårddiskar.

1 Problemet

Antag att du har en hårddisk med fler än 1024 cylindrar. Antag också att du har ett operativsystem som använder sig av BIOS. Då har du problem, eftersom det vanliga INT13-BIOS-gränssnittet för hårddisk-I/O använder ett 10-bitarsfält för att ange aktuell cylinder, så att cylinder 1024 och däröver är oåtkomliga.

Som tur är använder Linux sig inte av BIOS, så det är inget problem.

Ja, förutom två saker, då:

(1) Linux är inte startat när du startar datorn och kan därför inte rädda dig från BIOS-problem. Det har vissa konsekvenser för LILO och liknande multibootprogram.

(2) Det är nödvändigt för alla operativsystem som använder en och samma hårddisk att komma överens om var partitionerna ligger. Med andra ord, om du använder både Linux och, säg, DOS på samma hårddisk måste båda tolka partitionstabellen på samma sätt. Detta har vissa konsekvenser för Linux-kärnan och för fdisk.

Nedan finns en ganska detaljerad beskrivning av alla relevanta detaljer. Notera att jag använde källkoden för Linux version 2.0.8 som referens. Andra versioner kan skilja sig åt lite grann.

2 Boot-fasen

När datorn startas läser BIOSet sektor 0 (känt som MBR - Master Boot Record) från den första hårddisken (eller från diskett) och hoppar till koden som finns där – normalt någon bootstrap-laddare. Dessa små bootstrap-program som hittas där har typiskt inga egna drivrutiner för hårddiskar och använder BIOS-tjänster. Detta innebär att en Linux-kärna bara kan laddas om den i sin helhet ligger inom de 1024 första cylindrarna.

Detta problem är enkelt att lösa: se till att kärnan (och möjligen andra filer som används under uppstart, som till exempel map-filer till LILO), ligger på en partition som i sin helhet ligger inom de 1024 första cylindrarna på en hårddisk som BIOS kan komma åt – förmodligen innebär detta den första eller andra hårddisken.

En annan sak är att bootblocksladdaren och BIOS måste vara överens om vilken geometri hårddisken har. Det kan hjälpa att ge parametern `"linear"` till LILO. Mer information om det längre ner.

3 Hårddiskgeometri och partitioner

Om du har flera operativsystem på dina hårddiskar använder var och en av dem en eller flera partitioner. Om operativsystemen inte är överens om hur partitionerna ligger kan konsekvenserna bli katastrofala.

MBR innehåller en *partitionstabell* som beskriver var (de primära) partitionerna finns. Det finns 4 poster i tabellen för 4 primära partitioner, och varje post ser ut som

```
struct partition {
        char active;    /* 0x80: bootbar, 0: ej bootbar */
        char begin[3];  /* CHS för första sektorn */
        char type;
        char end[3];    /* CHS för sista sektorn */
        int start;      /* sektornummer (32-bitarstal, med början på 0) */
        int length;     /* antal sektorer (32-bitarstal) */
};
```

(där CHS står för cylinder/huvud/sektor).

Sålunda är informationen redundant: stället partitionen ligger på är givet både av de 24-bitars `begin`- och `end`-fälten och av de 32-bitars `start`- och `length`-fälten.

Linux använder bara `start`- och `length`-fälten och kan därför hantera partitioner som inte är större än 2^32 sektorer, det vill säga högst 2 TB stora partitioner. Det är hundra gånger större än hårddiskarna som är tillgängliga idag, så det kanske räcker för de nästkommande 80 åren, eller så.

Olyckligtvis använder INT13-BIOS-anropet CHS kodad som tre byte, med 10 bitar för antal cylindrar, 8 bitar för antal huvuden och 6 bitar för antal sektorer. Möjliga värden för antal cylindrar är 0-1023, möjliga värden för antal huvuden är 0-255 och möjliga värden för antal sektorer är 1-63 (ja, sektorer på ett spår räknas med början på 1, inte 0). Med dessa 24 bitar kan man adressera 8455716864 byte (7,875 GB), tvåhundra gånger större än hårddiskarna som fanns tillgängliga 1983.

Ännu mer olyckligt är att standardmässiga IDE-gränssnitt tillåter 256 sektorer/spår, 65536 cylindrar och 16 huvuden. Detta i sig själv tillåter åtkomst till 2^37 = 137438953472 byte (128 GB), men kombinerat med BIOS-begränsningarna (högst 63 sektorer och 1024 cylindrar) ger detta att bara 528482304 byte (504 MB) kan adresseras.

Detta är räcker inte för dagens hårddiskar, och folk får ta till alla sorters trick, både hårdvaru- och mjukvarumässiga.

4 Översättning och hårddiskhanterare

Ingen är intresserad av vad den riktiga" geometrin för en hårddisk är. Faktiskt är antalet sektorer per spår ofta variabelt – det finns fler sektorer per spår nära den yttre kanten på hårddisken – så det finns inget riktigt" antal sektorer per spår. För användaren är det bäst att betrakta en hårddisk som en linjär samling sektorer numrerade 0, 1, ..., och låta hårddiskkontrollern ta reda på var en given sektor finns på hårddisken.

Denna linjära numrering kallas för LBA. Den linjära adressen som hör till (c,h,s) för en hårddisk med geometrin (C,H,S) är c*H*S + h*S + (s-1). Alla SCSI-kontrollrar, och några IDE-kontrollrar, pratar LBA.

Om BIOSet konverterar 24-bitarsadressen (c,h,s) till LBA och skickar den till en kontroller som förstår LBA kommer 7,875 GB att vara adresserbart. Inte nog för alla sorters hårddiskar, men i alla fall en förbättring. Notera att CHS, så som det används av BIOS, här inte längre har någon koppling till "verkligheten".

Någonting liknande fungerar när kontrollern inte pratar LBA, men när BIOS känner till översättning. (Det indikeras ofta som Large" i BIOS-inställningarna.) Då kommer BIOS att presentera en geometri (C',H',S') för operativsystemet och använda (C,H,S) när den kommunicerar med diskkontrollern. Normalt är S = S', C' = C/N och H' = H*N, där N är den minsta exponenten till två som säkerställer att C' <= 1024 (så att så lite av kapaciteten som möjligt slösas bort i avrundning av C' = C/N). Åter igen, detta ger en adressrymd på 7,875 GB.

Om ett BIOS inte känner till Large" eller LBA" finns det mjukvarulösningar. Hårddiskhanterare som OnTrack eller EZ-Drive byter ut BIOS' rutiner för hårddiskhantering mot sina egna. Detta åstadkoms ofta genom att låta hårddiskhanterarens kod ligga i MBR och påföljande sektorer (OnTrack kallar denna kod DDO:

Dynamic Drive Overlay), så att den laddas innan operativsystemet. Det är därför man kan ha problem med att boota från diskett när en hårddiskhanterare har installerats.

Effekten är mer eller mindre likadan som med ett översättande BIOS – men hårddiskhanterare kan ställa till med mycket problem, speciellt när man kör flera olika operativsystem på samma hårddisk.

Linux stöder hårddiskhanteraren OnTrack sedan version 1.3.14 och EZ-Drive sedan version 1.3.29. Lite mer detaljer ges nedan.

5 Hårddisköversättning av kärnan för IDE-hårddiskar

Om Linux-kärnan detekterar närvaron av någon hårddiskhanterare på en IDE-hårddisk kommer den att försöka mappa om hårddisken på samma sätt som denna hårddiskhanterare skulle ha gjort, så att Linux ser samma hårddiskpartitionering som till exempel DOS med OnTrack eller EZ-Drive. Emellertid görs INGEN ommappning när en geometri specificeras på kommandoraden – så ett kommandoradsargument som "`hd=`*cylindrar*`,`*huvuden*`,`*sektorer*" skulle mycket väl kunna förstöra kompatibiliteten med en hårddiskhanterare.

Ommappningen görs genom att testa 4, 8, 16, 32, 64, 128, 255 huvuden (och hålla H*C konstant) tills antingen C <= 1024 or H = 255.

Detaljerna är som följer – titlarna för subkapitlen är strängarna som dyker upp i korresponderande bootmeddelanden. Här och överallt annars i denna text ges partitionstyperna med hexadecimaler.

5.1 EZD

EZ-Drive kan kännas igen genom att den första primära partitionen är av typ 55. Geometrin mappas om enligt beskrivning ovan, och partitionstabellen från sektor 0 slängs bort – istället läses partitionstabellen från sektor 1. Hårddiskblockens nummer ändras inte, men skrivningar till sektor 0 dirigeras om till sektor 1. Detta beteende kan ändras genom att kompilera om kärnan med `#define FAKE_FDISK_FOR_EZDRIVE 0` i *ide.c*.

5.2 DM6:DDO

Hårddiskhanteraren OnTrack (på den första hårddisken) känns igen genom att den första primära partitionen är av typ 54. Geometrin mappas om enligt beskrivning ovan, och hela disken skiftas 63 sektorer (så att den ursprungliga sektorn 63 blir sektor 0). Efteråt läses en ny MBR (med partitionstabell) in från sektor 0. Självklart är denna skiftning till för att göra plats för DDOn – det är därför det inte görs någon skiftning på resten av hårddiskarna.

5.3 DM6:AUX

Hårddiskhanteraren OnTrack (på resten av hårddiskarna) känns igen genom att den första primära partitionen är av typ 51 eller 53. Geometrin mappas om enligt beskrivning ovan.

5.4 DM6:MBR

En äldre version av hårddiskhanteraren OnTrack känns inte igen genom partitionstyp, utan genom en signatur. (Testa huruvida offseten som finns i byte 2 och 3 på MBRen inte är större än 430 och talet som finns på denna offset är lika med 0x55AA och följs av en udda byte.) Återigen, geometrin mappas om enligt ovan.

5.5 PTBL

Till sist finns ett test som försöker beräkna en översättning med hjälp av `start`- och `end`-fälten för de primära partitionerna: Om någon partition har start- och slutcylinder mindre än 256, och start- och slutsektor nummer 1 och 63, respektive, och sluthuvuden 31, 63 eller 127, så (eftersom det är brukligt att avsluta partitioner på en cylindergräns och eftersom IDE-gränssnittet använder högst 16 huvuden) antas det att en BIOS-översättning är aktiv och att geometrin är ommappad till att använda 32, 64 eller 128 huvuden, respektive. (Det finns kanske en brist här; *genhd.c* borde kanske inte ha testat de två mest signifikanta bitarna av cylindertalet?) Emellertid görs ingen ommappning när den nuvarande idén om hur geometrin ska vara inbegriper 63 sektorer per spår och åtminstone lika många huvuden (eftersom det då förmodligen innebär att en ommappning redan är gjord).

6 Konsekvenser

Vad betyder allt det här? För Linuxanvändare bara en sak: att de måste se till att LILO och fdisk använder rätt geometri där rätt" för fdisk är definierat som geometrin som används av de andra operativsystemen på samma hårddisk, och rätt" för LILO definieras som geometrin som möjliggör lyckad interaktion med BIOSet vid uppstart. (Normalt sett sammanfaller dessa.)

Hur känner fdisk till geometrin? Den frågar kärnan genom att använda ioctl:n `HDIO_GETGEO`. Men användaren kan själv, interaktivt eller på kommandoraden, bestämma vilken geometri som ska användas.

Hur känner LILO till geometrin? Den frågar kärnan genom att använda ioctl:n `HDIO_GETGEO`. Men användaren kan själv, med hjälp av parametern `"disk="`, bestämma vilken geometri som ska användas. Man kan också ge parametern `"linear"` till LILO, som då kommer att lagra LBA-adresser istället för CHS-adresser i sin map-fil och räkna ut vilken geometri som ska användas vid uppstart (genom att använda INT 13 funktion 8 för att fråga om hårddiskgeometrin).

Hur vet kärnan vad den ska svara? Jo, användaren kan explicit ha specificerat en geometri med en parameter som `hd=`*cyls*`,`*heads*`,`*secs*. Annars frågar kärnan hårdvaran.

6.1 IDE-detaljer

Låt mig utveckla mig. IDE-drivrutinen har fyra källor för information om geometrin. Den första (G_user) är den som användaren kan ha specificerat på kommandoraden. Den andra (G_bios) är BIOS' parametertabell (för fixa hårddiskar; bara för den första och den andra hårddisken) som läses innan bytet till 32-bitarsmod. Den tredje (G_phys) och fjärde (G_log) returneras av IDE-kontrollern som ett svar på kommandot IDENTIFY – de är de fysiska" och nuvarande logiska" geometrierna.

Å andra sidan behöver drivrutinen två värden för geometrin: för det första G_fdisk, som returneras av en `HDIO_GETGEO`-ioctl, och för det andra G_used, som faktiskt används för I/O. Både G_fdisk och G_used initialiseras till G_user om den är given, till G_bios om informationen finns tillgänglig enligt CMOS, och annars till G_phys. Om G_log verkar vara rimlig sätts G_used till det. Annars, om G_used inte verkar rimlig och G_phys ser rimlig ut, sätts G_used till G_phys. Här betyder rimlig" att antalet huvuden ligger i intervallet 1-16.

Med andra ord: kommandoraden har högre prioritet än BIOSet och bestämmer vad fdisk ser, men om det specificerar en översatt geometri (med fler än 16 huvuden) kommer det som returneras från IDENTIFY-kommandot att användas för kärn-I/O.

6.2 SCSI-detaljer

Situationen för SCSI är något annorlunda, eftersom SCSI-kommandona redan använder logiska blocknummer, så en geometri" är helt irrelevant för I/O i praktiken. Formatet för partitionstabellen är emellertid fortfarande samma, så fdisk måste uppfinna en geometri och använder `HDIO_GETGEO` här också – faktiskt skiljer inte fdisk på IDE- och SCSI-hårddiskar. Som man kan förstå från den detaljerade beskrivningen nedan, uppfinner de olika drivrutinerna en något annorlunda geometri. Helt klart en sagolik röra.

Om du inte använder DOS eller så, undvik alla inställningar rörande utökad översättning och använd bara 64 huvuden, 32 sektorer per spår (vilket ger trevliga 1 MB per cylinder), om möjligt, så att inga problem uppstår när du flyttar hårddisken från en kontroller till en annan. Vissa drivrutiner för SCSI-hårddiskar (aha152x, pas16, ppa, qlogicfas, qlogicisp) är så måna om att bibehålla DOS-kompatibilitet att de inte tillåter ett system som bara kör Linux att använda mer än 8 GB. Detta är en bugg.

Vad är den riktiga geometrin? Det lättaste svaret är att det inte finns någon sådan. Och om det fanns skulle du inte vilja vet något om den och säkerligen ALDRIG NÅGONSIN tala om den för fdisk eller LILO eller kärnan. Det är endast relevant för SCSI-kontrollern och hårddisken. Låt mig upprepa: bara dumma personer talar om den riktiga geometrin för en SCSI-hårddisk för fdisk/LILO/kärnan.

Men om du är nyfiken och insisterar kan du fråga hårddisken själv. Det finns det viktiga kommandot READ CAPACITY som ger den totala storleken på hårddisken och det finns kommandot MODE SENSE som i Rigid Disk Drive Geometry Page" (sida 04) ger antalet cylindrar och huvuden (detta är information som inte går att ändra), och Format Page" (sida 03) ger antal byte per sektor och antal sektorer per spår. Det sistnämnda antalet är typiskt beroende av notchen och antalet sektorer per spår varierar – de yttre spåren har fler sektorer än de inre. Linuxprogrammet scsiinfo kan ge den här informationen. Det finns många detaljer och komplikationer och det står klart att ingen (inte ens operativsystemet) vill använda denna information. Dessutom: så länge man enbart bryr sig om fdisk och LILO får man typiskt svar som C/H/S=4476/27/171 – värden som inte kan användas av fdisk eftersom partitionstabellen endast reserverar 10 respektive 8 respektive 6 bitar för C/H/S.

Så var får kärnanropet `HDIO_GETGEO` sin information från? Jo, antingen från SCSI-kontrollern eller genom att göra en kvalificerad gissning. Några drivrutiner verkar tro att vi vill känna till "verkligheten", men självklart vill vi bara veta vad FDISK i DOS eller OS/2 (eller Adaptecs AFDISK, etc) kommer att använda.

Notera att Linux' fdisk behöver talen H och S (för huvuden och sektorer) för att kunna konvertera LBA-sektornummer till C/H/S-adresser, men talet C (cylindrar) spelar ingen roll i den här konverteringen. Några drivrutiner använder (C,H,S) = (1023,255,63) för att signalera att hårddiskkapaciteten är minst 1023*255*63 sektorer. Detta är olyckligt, eftersom det inte avslöjar den riktiga storleken, vilket kommer att begränsa användarna av de flesta fdisk-versionerna till att använda 8 GB av sina hårddiskar – en riktig begränsning i dessa dagar.

I beskrivningen nedan står M för den totala hårddiskkapaciteten och C, H, S för antalet cylindrar, huvuden och sektorer per spår. Det räcker att ge H, S om vi betraktar C som definierad av M / (H*S).

Standardmässigt är H=64, S=32.

aha1740, dtc, g_NCR5380, t128, wd7000:

H=64, S=32.

aha152x, pas16, ppa, qlogicfas, qlogicisp:

H=64, S=32 om inte C > 1024, då H=255, S=63, C = min(1023, M/(H*S)). (Alltså trunkeras C och H*S*C är inte en approximation för hårddiskkapaciteten M. Detta gör de flesta fdisk-versioner förvirrade.) Koden i *ppa.c* använder M+1 istället för M och säger på grund av en bugg i *sd.c* att M är ett för lite.

advansys:

H=64, S=32 om inte C > 1024 och "> 1 GB" är inställt i BIOS, då H=255, S=63.

aha1542:

Fråga kontrollern vilket av två möjliga sätt att översätta på som ska användas och använd antingen H=255, S=63 eller H=64, S=32. I det första fallet blir det ett boot-meddelande: äha1542.c: Using extended bios translation".

aic7xxx:

H=64, S=32 om inte C > 1024 och antingen boot-parametern "extended" gavs eller att biten "extended" var satt i SEEPROM eller BIOS, då H=255, S=63.

buslogic:

H=64, S=32 om inte C >= 1024, och om utökad översättning är inställt på kontrollern, så är H=128, S=32 om M < 2^22, annars H=255, S=63. Efter att detta val har gjorts läses partitionstabellen och om värdet endH=H-1 syns för någon av de tre möjligheterna (H,S) = (64,32), (128,32), (255,63), blir det ett boot-meddelande: Ädopting Geometry from Partition Table".

fdomain:

Ta reda på geometriinformationen från parametertabellen för hårddiskar i BIOS eller läs partitionstabellen och använd H=endH+1, S=endS för den första partitionen, ifall den inte var tom, annars: använd H=64, S=32 för M < 2^21 (1 GB), H=128, S=63 för M < 63*2^17 (3,9 GB) och H=255, S=63.

in2000:

Använd den första av (H,S) = (64,32), (64,63), (128,63), (255,63) som gör att C <= 1024. I det sista fallet, trunkera C vid 1023.

seagate:

Läs C,H,S från hårddisken. (Iiiick!) Om C eller S är för stor, sätt S=17, H=2 och dubbla H tills C <= 1024. Detta innebär att H kommer att sättas till 0 om M > 128*1024*17 (1,1 GB). Detta är en bugg.

ultrastor och u14_34f:

En av tre mappningar ((H,S) = (16,63), (64,32), (64,63)) används beroende på kontrollerns mappningsmod.

Om drivrutinen inte specificerar geometrin använder vi en kvalificerad gissning genom att använda partitionstabellen eller genom att använda den totala hårddiskkapaciteten.

Kolla på partitionstabellen. Eftersom, per konvention, partitioner slutar på en cylindergräns kan vi, givet `end = (endC,endH,endS)` för vilken partition som helst, helt enkelt sätta H = `endH+1` och S = `endS`. (Kom ihåg att sektorerna räknas från 1.) Mer precist görs följande: Om det finns en icke-tom partition, välj den partition med störst `beginC`. Titta på `end+1` för den partitionen, beräknat genom att både addera `start` och `length` och genom att anta att partitionen slutar på en cylindergräns. Om båda värdena stämmer överens eller om `endC` = 1023 och `start+length` är en heltalsmultipel av `(endH+1)*endS`, anta att partitionen faktiskt var placerad på en cylindergräns och sätt H = `endH+1` och S = `endS`. Om detta inte fungerar, antingen på grund av att det inte finns några partitioner eller på grund av att de har konstiga storlekar, kolla bara på hårddiskkapaciteten M. Algoritm: sätt H = M/(62*1024) (avrundat uppåt), S = M/(1024*H) (avrundat uppåt), C = M/(H*S) (avrundat neråt). Detta ger en (C,H,S) med C högst 1024 och S högst 62.

7 Linux 8 GB-begränsning för IDE

Linux IDE-drivrutin hämtar geometrin och hårddiskkapaciteten (och många andra saker) genom att använda ett ATA IDENTIFY-anrop. Tills nyligen trodde inte drivrutinen på det returnerade värdet från lba_capacity om det var mer än 10 % större än kapaciteten beräknad genom C*H*S. Nyare Quantum Bigfoot 12 GB-hårddiskar returnerar emellertid C=16383, H=16, S=63, vilket ger totalt 16514064 sektorer (7,8 GB) men rapporterar lba_capacity som 23547888 sektorer (det vill säga 11,2 GB, C=23361).

Nyare Linuxkärnor (2.0.34pre14, 2.1.90) känner till detta och beter sig rätt. Om du har en äldre Linuxkärna och inte vill uppgradera, och kärnan bara ser 8 GB av en mycket större hårddisk, kan du testa att ändra rutinen `lba_capacity_is_ok` i `/usr/src/linux/drivers/block/ide.c` till någonting likt

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

För en mer försiktig patch, se 2.1.90.