Lilo mini-Howto

Cameron Spitzer (cls@truffula.sj.ca.us), Alessandro Rubini (rubini@linux.it). Vertaling: Jasper Aukes (jasper@nl.linux.org).
v2.02, 16 Augustus 1998, vertaling 11 April 1999

LILO is de meest gebruikte **Li**nux **Lo**ader voor de x86 versie van Linux; Ik zal het Lilo noemen in plaats van LILO omdat ik niet van hoofdletters houd. Dit document beschrijft een paar veel voorkomende Lilo installaties. Het is bedoeld om een aanvulling te zijn op de Lilo gebruiksaanwijzing. Ik denk dat de voorbeelden informatief zijn, ook al is jouw setup verschillend van die van mij. Ik hoop dat dit je moeite bespaart. Omdat de Lilo documentatie al erg goed is, verwijs ik voor details naar /usr/doc/lilo*

Inhoudsopgave

1 Inleiding

Alhoewel de documentatie in Lilo's bronbestanden (die geinstalleerd zijn in `/usr/doc/lilo`-versie) erg begrijpelijk is, hebben de meeste Linux gebruikers problemen met het maken van een eigen `/etc/lilo.conf` bestand. Dit document is bedoeld om hen te ondersteunen door ze minimale informatie te geven en door het tonen van een vijftal eenvoudige installatie-voorbeelden:

- Het eerste voorbeeld is de klassieke "Linux en anders" installatie.
- De volgende toont hoe Lilo te installeren is op een harddisk die aangesloten is als `/dev/hdc` die boot als `/dev/hda`. Normaal gesproken is dit nodig als je Linux installeert op een nieuwe disk vanaf je draaiende systeem. Het laat ook zien hoe van een SCSI disk op te starten is als je BIOS modern genoeg is.

- Het derde voorbeeld toont hoe je een Linux systeem boot waarvan de root partitie niet toegankelijk is voor het BIOS.
- Het volgende voorbeeld bestand wordt gebruikt om erg grote disks aan te spreken, die noch door het BIOS, noch door DOS eenvoudig kan worden aangesproken. (deze is een beetje verouderd).
- Het laatste voorbeeld laat zien hoe een beschadigde disk te herstellen is als de schade veroorzaakt is door een ander OS te installeren.

De laatste drie voorbeelden zijn van Cameron, `cls@truffula.sj.ca.us`, die het originele document schreef. Alessandro `rubini@linux.it`, die momenteel het document onderhoudt, draait enkel Linux en kan ze niet zelf verifieren of updaten. Onnodig te zeggen dat commentaar welkom is.

2 Achtergrond Informatie en Standaard Installatie

Als Lilo het systeem opstart, gebruikt het BIOS calls om de Linux kernel van de disk (IDE drive, floppy of wat dan ook) te laden. Daarom moet de kernel staan op een plek waar het BIOS bij kan.

Tijdens het opstarten is Lilo niet in staat om filesystem gegevens te lezen en iedere padnaam die in `/etc/lilo.conf` staat wordt gevonden tijdens het installeren (als je */sbin/lilo* draait). Tijdens het installeren bouwt het programma de tabellen waarin de sectoren staan die gebruikt worden door de bestanden die het operating systeem laden. Een consequentie hiervan is, dat al deze bestanden in een partitie moeten staan die toegankelijk is voor het BIOS (de bestanden staan meestal in de `/boot` directory, wat betekent dat alleen de root partitie van je Linux systeem door het BIOS benaderbaar moet zijn).

Een andere consequentie van het BIOS gebaseerd zijn is dat je de lader moet her-installeren (d.w.z, */sbin/lilo* draaien) telkens als je de Lilo setup wijzigt. Telkens als je een nieuwe kernel compileert en hem over je oude image heen zet, moet je Lilo opnieuw installeren.

2.1 Waar Moet Ik Lilo Installeren?

De `boot=` opdracht in `/etc/lilo.conf` vertelt Lilo waar het de primaire boot loader moet neerzetten. In het algemeen kun je hier het master boot record (`/dev/hda`) of de root partitie van je Linux installatie (is normaal gesproken `/dev/hda1`) aangeven.

Als je nog een ander operating systeem op je harde schijf hebt geinstalleerd, kun je Lilo beter in de root partitie zetten in plaats van het MBR. In dat geval moet je de partitie als “bootable” markeren door het “a” commando van *fdisk* of het “b” commando van *cfdisk* te gebruiken. Als je het master boot record niet overschrijft is het makkelijker om Linux en Lilo te de-installeren als dat nodig is.

2.2 Hoe Moet Ik mijn IDE Harddisks Configureren?

Persoonlijk gebruik ik geen LBA of LARGE instellingen in het BIOS (maar ik draai enkel Linux); het is een enorm knullige oplossing, veroorzaakt door ontwerpfouten in de PC wereld. Hierdoor moet de kernel in de eerste 1024 cylinders staan, maar dat is geen probleem zolang je je harddisks partitioneert en je de root partitie klein houdt (wat je toch al moet doen).

Als je harddisk al een ander operating systeem bevat kun je de BIOS instellingen niet aanpassen, anders draait het oude systeem niet meer. Alle recente Lilo distributies kunnen met LBA en LARGE disk instellingen overweg.

Merk op dat de `"linear"variabele in /etc/lilo.conf kan helpen tegen afmetings problemen. De variabele instrueert Lilo om lineaire sector adressen te gebruiken in plaats van` sector/head/cylinder tupels. De conversie naar 3D adressen is vertraagd naar run-time en maakt zodoende de setup beter bestand tegen afmetings problemen.

Als je meer dan een harddisk hebt en sommigen hiervan alleen door Linux gebruikt worden, maar niet betrokken zijn bij het boot proces, kan je je BIOS vertellen dat ze niet geinstalleerd zijn. Je systeem zal sneller opstarten en Linux zal de disks zelf heel snel detecteren. Ik wissel vaak disks in mijn computers, maar kom nooit aan de BIOS configuratie.

2.3 Hoe Kan Ik Kiezen Tijdens het Opstarten?

Als je de Lilo prompt ziet, kan je door op de <Tab> toets te drukken een lijst met mogelijke keuzes te voorschijn halen. Als Lilo niet geconfigureerd is om interactief te zijn, moet je de <Alt> of de <Shift> toets ingedrukt houden voordat de “LILO” melding verschijnt.

Als je kiest om een Linux kernel op te starten, kun je command-line argumenten toevoegen achter de naam van het systeem dat je kiest. De kernel accepteert vele command-line argumenten. Alle argumenten worden opgesomt in de “BootPrompt-HOWTO” van Paul Gortmaker en ik zal ze hier niet herhalen. Een paar command-line argumenten zijn echter bijzonder belangrijk en zullen hier beschreven worden:

- root=: je kunt de Linux kernel vertellen als root een andere partitie te mounten dan degene die in lilo.conf staat. Mijn systeem bijvoorbeeld heeft een hele kleine partite met een minimale Linux installatie erop en ik kan het systeem ermee opstarten als ik per ongeluk mijn root partitie vernield heb.
- init=: kernel versie 1.3.43 en nieuwer kan in plaats van /sbin/init, het gespecificeerde commando uitvoeren. Als je vervelende problemen tijdens het opstarten ondervindt, kun je het kale systeem benaderen door init=/bin/sh op te geven (als je dan de shell prompt krijgt zul je hoogst waarschijnlijk zelf je disks moeten mounten: probeer “mount -w -n -o remount /; mount -a” en onthoud om “umount -a” te doen voordat je de computer uit doet.
- Een getal: door een getal op te geven op de kernel command-line, geef je een specifiek run-level op aan *init* (normaal gesproken is dat 3 of 2, afhankelijk van de distributie die je gebruikt). Ga de *init* documentatie, /etc/inittab en /etc/rc*.d na om dit verder uit te zoeken.

2.4 Hoe Kan Ik Lilo Verwijderen?

Als Lilo de boot sector overschrijft, bewaart het een kopie in /boot/boot.*xxyy*, waar *xxyy* de major en minor getallen van het device zijn in hexadecimale notatie. Je kunt de major en minor getallen van je disk of partitie zien door “ls -l /dev/*device*” te draaien. Bijvoorbeeld, de eerste sector van /dev/hda (major 3, minor 0) zal bewaard worden in /boot/boot.0300, als je Lilo installeert op /dev/fd0 wordt /boot/boot.0200 aangemaakt en als je Lilo op /dev/sdb3 (major 8, minor 19) installeert, wordt /boot/boot.0819 aangemaakt. Merk op dat Lilo het bestand niet aan zal maken als er al een is. Je hoeft je dus geen zorgen te maken over de reserve kopie als je Lilo her-installeert (bijvoorbeeld nadat je een nieuwe kernel hebt gecompileerd). De reserve kopieen in /boot/ zijn altijd een momentopname van de situatie voor de installatie van Lilo.

Als je ooit Lilo wilt verwijderen (bijvoorbeeld in het jammerlijke geval dat je Linux wilt de-installeren), dan hoef je alleen de originele boot sector te herstellen. Als Lilo geinstalleerd is op /dev/hda, dan hoef je alleen "dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1" te doen (Ik doe zelf gewoon "cat /boot/boot.0300 > /dev/hda", maar dat is niet veilig, omdat het tevens de originele partitie tabel herstelt, terwijl je die intussen ook aangepast zou kunnen hebben). Dit commando is veel eenvoudiger te draaien dan "fdisk /mbr" in een DOS shell te draaien: het staat je toe Linux netjes van een disk te verwijderen zonder ook maar iets anders op te starten dan Linux. Nadat Lilo verwijderd is kunnen met Linux' *fdisk* alle Linux partities vernietigd worden (DOS' *fdisk* kan dat niet)

Als je Lilo op je root partitie hebt geinstalleerd (bv. /dev/hda2), hoeft je niets speciaals te doen om Lilo te verwijderen. Draai Linux' *fdisk* om de Linux partities te verwijderen uit de partitie tabel. Je moet ook de DOS partitie als "bootable" markeren.

3 De Eenvoudige Configuratie

De meeste Lilo installaties hebben een configuratie file zoals hieronder:

```
boot = /dev/hda   # of je root partitie
delay = 10        # vertraging, in tienden van een sec. (zodat je kunt kiezen)
vga = 0           # optioneel. Gebruik "vga=1" om 80x50 te krijgen
#linear           # probeer "linear" bij afmetings problemen.

image = /boot/vmlinux  # je zImage bestand
  root = /dev/hda1     # je root partitie
  label = Linux        # of een willekeurige hippe naam
  read-only            # mount root 'alleen lezen'

other = /dev/hda4   # je dos partitie, als die er is
  table = /dev/hda  # de huidige partitie tabel
  label = dos       # of elke willekeurige saaie naam
```

Je kunt meerdere "image" en "other" secties gebruiken als je wilt. Het is niet ongebruikelijk om verschillende kernel images te configureren in je *lilo.conf*, ten minste, als je de kernel ontwikkelingen bijhoudt.

3.1 Hoe met Grote Kernels te Werken

Als je een "zImage" kernel compileert en deze te groot is om in een halve megabyte te passen (wat gebruikelijk is voor de nieuwe 2.2 kernels), moet je een " big zImage" maken: "make bzImage". Om een grote kernel image op te starten hoef je verder niets speciaals te doen, maar je hebt versie 18 of hoger van Lilo nodig. Als je installatie ouder is moet je je Lilo pakket upgraden.

3.2 Andere Informatiebronnen

In aanvulling op de Lilo documentatie is er een aantal mini-howto's die handig kunnen zijn. Deze zijn allemaal "Linux+*foobarOS*" genoemd, ze gaan over het naast elkaar bestaan van Linux en een ander(e) besturingssyste(em/men). Daarnaast beschrijft "Multiboot-with-LILO" hoe de verschillende Windows smaken naast Linux kunnen bestaan.

4 "hdcäls "hda"Laten Booten en "bios="Gebruiken

Lilo staat toe de kernel image op de ene disk te zetten en de BIOS te instrueren hem van een andere disk te halen. Het is bijvoorbeeld normaal voor mij om Linux te installeren op een disk die ik aansluit op hdc (master disk van de secundaire controller) en hem op te starten als 'standalone' systeem op de primary IDE controller van een andere computer. Ik heb de installatie floppy gekopieerd naar een kleine partitie, zodat ik hem *chroot* kan draaien in een virtueel console om hdc te installeren terwijl ik het systeem voor andere doeleinden gebruik.

Het *lilo.conf* bestand dat ik gebruikte om Lilo te installeren zag er als volgt uit:

```
# Dit file moet gebruikt worden voor een systeem dat vanaf /dev/hdc draait
boot = /dev/hdc   # overschrijf MBR van hdc
disk = /dev/hdc   # vertel hoe hdc eruit ziet:
   bios = 0x80    #  de bios zal het als de eerste drive zien
delay = 0
vga = 0

image = /boot/vmlinux  # dit staat op /dev/hdc1
  root = /dev/hda1     # maar is tijdens het opstarten hda1
  label = Linux
  read-only
```

Dit configuratie bestand moet door Lilo gelezen worden die van /dev/hdc1 gedraaid wordt. De Lilo tabellen die geschreven worden in de boot sector (/dev/hdc) moeten refereren aan bestanden in /boot (momenteel geinstalleerd als hdc); die files zullen worden benaderd via hda wanneer deze disk geboot zal worden als standalone systeem.

Ik noem dit configuratie bestand /mnt/etc/lilo.conf.hdc. (/mnt is waar hdc gemount wordt tijdens de installatie) Ik installeer Lilo door "cd /mnt; chroot . sbin/lilo -C /etc/lilo.conf.hdc" aan te roepen. Bekijk de manual-page van *chroot* als dit magisch oogt.

De "bios=" opdracht in lilo.conf wordt gebruikt om Lilo te vertellen wat het BIOS over je apparaten weet. BIOS calls identificeren floppy drives en harddisks met een nummer: 0x00 en 0x01 selecteren de floppy drives, 0x80 en volgend selecteren harddisks (oude BIOSsen kunnen maar twee disks aan). De betekenis van "bios = 0x80" in het vorige voorbeeld betekent dus "gebruik 0x80 in je BIOS calls voor /dev/hdc".

Deze Lilo opdracht kan handig zijn in andere situaties, bijvoorbeeld wanneer je BIOS van SCSI disks kan opstarten in plaats van IDE disks. Als IDE en SCSI beide aanwezig zijn, kan Lilo niet uitmaken naar welke van de twee 0x80 verwijst omdat de gebruiker dat kan kiezen in de BIOS configuratie menu's en de BIOS niet benaderbaar is terwijl Linux draait.

Standaard gaat Lilo ervan uit dat IDE drives als eerste door de BIOS worden ingedeeld, maar dit kan worden overschreven door gebruik te maken van ongeveer de volgende instructies in /etc/lilo.conf:

```
disk = /dev/sda
  bios = 0x80
```

5 Lilo Gebruiken Terwijl het BIOS de Root Partitie Niet Ziet

Ik heb twee IDE disks en een SCSI disk. De SCSI disk wordt niet door het BIOS gedetecteerd. De Linux Loader, Lilo, gebruikt BIOS calls en kan alleen de drives zien die het BIOS kan zien. Mijn achterlijke AMI BIOS wil alleen opstarten van "A:öf "C:"Mijn root filesysteem staat op een partitie op de SCSI disk.

De oplossing bestaat uit het opslaan van de kernel, het map bestand en ketting lader in een Linux partitie op de eerste IDE disk. Merk op dat het niet noodzakelijk is om je kernel op je root partitie te hebben staan.

De tweede partitie van mijn eerste IDE disk (/dev/hda2, de Linux partitie die gebruikt wordt om het systeem op te starten) is gemount onder /u2. Hier komt het /etc/lilo.conf bestand dat ik gebruikte:

```
#  Installeer Lilo op het Master Boot Record
#  op de eerste IDE disk.
#
boot = /dev/hda
#  /sbin/lilo (het installatie programma) kopieert het Lilo boot record
#  van het volgende bestand naar de MBR locatie.
install = /u2/etc/lilo/boot.b
#
#  Ik heb een uitgebreid boot menu geschreven. Lilo vindt het hier.
message = /u2/etc/lilo/message
#  Het installatieprogramma zal het volgende bestand gaan aanmaken. Het vertelt
#  de boot-loader waar de blokken van de kernel staan.
map = /u2/etc/lilo/map
compact
prompt
#  Wacht 10 seconde, start daarna de 1.2.1 kernel als standaard.
timeout = 100
#  De kernel is opgeslagen op een plaats waar het BIOS het kan zien door:
#      cp -p /usr/src/linux/arch/i386/boot/zImage /u2/z1.2.1
image = /u2/z1.2.1
        label = 1.2.1
#  Lilo vertelt de kernel om de eerste SCSI partitie te mounten
#  als root partitie. Het BIOS hoeft dit niet te kunnen zien.
        root = /dev/sda1
#  Deze partitie zal gecontroleerd en opnieuw gemount worden door /etc/rc.d/rc.S
        read-only
#  Ik heb een oude Slackware kernel bewaard voor het geval ik een kernel heb
#  gebouwd die niet werkt. Ik heb deze echt een keer nodig gehad.
image = /u2/z1.0.9
        label = 1.0.9
        root = /dev/sda1
        read-only
#  Mijn DR-DOS 6 partitie.
```

```
other = /dev/hda1
        loader=/u2/etc/lilo/chain.b
        label = dos
        alias = m
```

6 Erg Grote Disks Benaderen Terwijl het BIOS dat Niet Kan

Het systeem op kantoor heeft een 1GB IDE disk. Het BIOS kan alleen de eerste 504 MB van de IDE disk zien. (Waar MB 2**10 bytes betekent, niet 10**6 bytes.) Ik heb MS-DOS op een 350 MB partitie /dev/hda1 en mijn Linux root op een 120 MB partitie /dev/hda2.

MS-DOS kon zichzelf niet goed installeren toen de disk nieuw was. Novell DOS 7 had hetzelfde probleem. Gelukkig was "Options by IBM"vergeten om de "OnTrack"diskette in de doos van de disk te doen. De disk had vergezeld moeten zijn van een produkt genaamd "OnTrack Disk Manager."Als je enkel MSDOS hebt, moet je het denk ik gebruiken.

Ik maakte dus een partitie tabel met Linux' fdisk. MSDOS-6.2 weigerde zich te installeren in /dev/hda1. Het zei iets als "deze uitgave van MS-DOS is voor nieuwe installaties. Je computer heeft al MS-DOS dus je hebt een upgrade uitgave van je verkoper nodig." In feite was de disk gloednieuw.

Wat een zootje! Ik draaide Linux' fdisk opnieuw en verwijderde partitie 1 van de tabel. Dit stelde MS-DOS 6.2 tevreden die doorging met het zelf aanmaken van exact dezelfde partitie 1 die ik zojuist zelf verwijderd had. MS-DOS 6.2 schreef zijn Master Boot Record op de disk, maar kon niet opstarten.

Gelukkig had ik een Slackware kernel op een floppy (aangemaakt door het Slackware installatie programma "setup"), dus ik startte Linux op en schreef Lilo over MS-DOS' defecte MBR. Dit werkt. Hier is het /etc/lilo.conf bestand dat ik gebruikte:

```
boot = /dev/hda
map = /lilo-map
delay = 100
ramdisk = 0             # Schakelt ramdisk uit in Slackware kernel
timeout = 100
prompt
disk = /dev/hda         # BIOS ziet slechts eerste 500 MB.
   bios = 0x80          # specificeert de eerste IDE disk.
   sectors = 63         # haal de cijfers uit de documentatie van je disk.
   heads = 16
   cylinders = 2100
image = /vmlinuz
  append = "hd=2100,16,63"
  root = /dev/hda2
  label = linux
  read-only
  vga = extended
other = /dev/hda1
  label = msdos
  table = /dev/hda
  loader = /boot/chain.b
```

Nadat ik deze systemen had geinstalleerd, verifieerde ik dat de partitie de zImage, boot.b, map, chain.b bevatte en begreep ik dat de bestanden gebruik kunnen maken van een MSDOS bestandssysteem zolang ze maar niet "gestackerdöf "gedoublespaced"zijn.

Ik heb ook geleerd dat "OnTrackëen partitie tabel zou hebben geschreven die begon met een tiental bytes verderop op de drive, in plaats van aan het begin en het is mogelijk om de Linux driver zo om te schrijven dat het dit probleem ook omzeilt. Maar installeren zou onmogelijk geweest zijn met de voorgecompileerde Slackware kernel. Uiteindelijk zond IBM me een "OnTrack"diskette. Ik belde OnTrack's technische ondersteuning. Ze vertelden me dat Linux onbruikbaar is omdat Linux het BIOS niet gebruikt. Ik heb de diskette weggegeven.

7 Opstarten met een Rescue Floppy

Vervolgens installeerde ik Windows-95 op het systeem van kantoor. Ik gooide mijn fraaie Lilo MBR weg, maar liet mijn Linux partities in stand. Kernels doen er te lang over om van floppy te starten dus ik maakte een floppy met een werkende Lilo setup erop die mijn kernel van IDE kon opstarten.

Ik maakte de Lilo floppy alsvolgt:

```
  fdformat /dev/fd0H1440      #  leg tracks op een nieuwe diskette
  mkfs -t minix /dev/fd0 1440 #  maak een minix filesysteem aan
  mount /dev/fd0 /mnt         #  mount in het standaard tijdelijke mount point
  cp -p /boot/chain.b /mnt    #  kopieer de chain loader
  lilo -C /etc/lilo.flop      #  installeer Lilo en de map op de diskette.
  umount /mnt
```

Merk op dat de diskette gemount moet zijn als je het installatie programma draait zodat Lilo zijn map bestand netjes kan wegschrijven.

Dit is het /etc/lilo.flop bestand. Het is bijna dezelfde als de vorige:

```
#  Maakt een floppy dat de kernels van HD kan opstarten.
boot = /dev/fd0
map = /mnt/lilo-map
delay = 100
ramdisk = 0
timeout = 100
prompt
disk = /dev/hda     # 1 GB IDE, BIOS ziet alleen de eerste 500 MB.
   bios=0x80
   sectors = 63
   heads = 16
   cylinders = 2100
image = /vmlinuz
  append = "hd=2100,16,63"
  root = /dev/hda2
  label = linux
  read-only
  vga = extended
other = /dev/hda1
```

```
  label = msdos
  table = /dev/hda
  loader = /mnt/chain.b
```

Eindelijk had ik MS-DOS 6.2, maar ik wilde niet meer aan mijn eerste drive komen. Ik voegde een SCSI controller toe met een drive, maakte er een msdos filesysteem met Linux' mkdosfs en Windows ziet het als "D:". Maar natuurlijk kan MSDOS niet opstarten vanaf D:. Dit is echter geen probleem als je Lilo hebt. Ik voegde het volgende toe aan lilo.conf van Voorbeeld 2.

```
other = /dev/sda1
  label = d6.2
  table = /dev/sda
  loader = /boot/any_d.b
```

Met deze aanpassing draait MSDOS-6.2 en denkt dat het op C: is en Windows op D:.