Linux LILO mini-HOWTO

Cameron Spitzer (cls@truffula.sj.ca.us), Alessandro Rubini (rubini@linux.it) und Horst Knobloch (hknobloch@odn.de)
v2.1-2, 9. Januar 1998

Diese Dokument beschreibt einige typische LILO-Installationen. Es soll eine zusätzliche Hilfe zum LILO User's Guide sein. Ich glaube, die Beispiele sind informativ, auch bei Konstellationen, die nicht gleich der von mir benutzten sind. Ich hoffe, daß sie helfen, Probleme zu vermeiden.

Inhaltsverzeichnis

1 Einleitung

Trotz der umfassenden Dokumentation, die den LILO Sourcen beiliegt und in `/usr/doc/lilo`-version zu finden ist, bereitet es den meisten Linux Benutzern Probleme ein eigenes `/etc/lilo.conf` File zu erstellen. Dieses Dokument ist gedacht, diese Benutzer zu unterstützen, indem es nur die notwendigsten Informationen erläutert und fünf Beispiel-Installationen zeigt:

- Das erste Beispiel ist die klassische Installation »Linux und anderes Betriebssystem«.
- Das nächste zeigt wie man Lilo auf einer Festplatte installiert, welche mit `/dev/hdc` angesprochen wird, aber dann als `/dev/hda` booten soll. Dies wird normalerweise benötigt, wenn man Linux von einem laufenden System auf eine neue Platte installiert.
- Das dritte Beispiel zeigt, wie man Linux bootet, wenn die Root-Partition nicht vom BIOS erreichbar ist.

- Das nächste Beispiel ist gedacht um den Zugriff auf große Platten zu demonstrieren, auf die weder das BIOS noch DOS einfach zugreifen kann. (Diese Beispiel ist etwas veraltet.)
- Das letzte Beispiel zeigt, wie man eine beschädigte Platte restauriert (der Schaden wurde durch die Installation eines anderen Betriebssystems verursacht).

Die letzten drei Beispiele sind von Cameron Spitzer (`cls@truffula.sj.ca.us`), welcher das Originaldokument schrieb. Da ich als der gegenwärtige Kümmerer nichts anderes als Linux benutzte, kann ich diese Beispiele weder prüfen noch anpassen.

1.1 Copyright

2 Hintergrund Information und Standard Installation

Wenn Lilo das System bootet, kann er nur die Daten-Sektoren laden, die über das BIOS erreichbar sind. Jeder Pfadname in `/etc/lilo.conf` wird zum Installationszeitpunkt ausgewertet, d.h. beim Aufruf von `lilo`. Dies ist auch der Zeitpunkt, bei dem das Programm die Zuordnungs-Tabelle (map) erstellt. Diese Tabelle gibt an, welche Sektoren von den Dateien benutzt werden. Daraus folgt als Konsequenz, daß die Dateien in einer Partition liegen müssen, welche über das BIOS ansprechbar ist. Außerdem muß jedesmal der Lader neu-installiert werden (d.h. Aufruf von `lilo`), sobald man die Files ändert. Nach jeder Kernel-Kompilierung und erfolgreicher Installation ist auch Lilo neu zu installieren.

2.1 Wohin Lilo installieren?

Die `boot =` Anweisung in `/etc/lilo.conf` sagt dem Lilo, wohin er den Haupt-Boot-Lader zu installieren hat. Im allgemeinen kann man den Master-Boot-Record (z.B. `/dev/hda`) oder die Root-Partition der Linux Installation angeben (z.B. `/dev/hda1` oder `/dev/hda2`).

Wenn ein anderes Betriebssystem auf der Platte installiert ist, sollte man Lilo besser in der Root-Partition unterbringen. In diesem Fall ist diese Partition als »bootable« (bootbar) zu markieren. Das erreicht man mit dem »a« Kommando von `fdisk` oder mit dem »b« Kommando von `cfdisk`. Es ist einfacher Lilo und Linux (bei Bedarf) zu entfernen, wenn Lilo nicht im *Master Boot Record* (MBR) installiert wird.

2.2 Wie IDE-Platten konfigurieren?

Da ich nur Linux benutze, benötige ich die LBA- oder LARGE-Einstellung im BIOS nicht. Diese Einstellungen sind fürchterliche Krücken um Design-Schwächen der PCs zu umgehen. Diese fordern, daß der Kernel in den ersten 1024 Zylindern liegen muß; das ist kein Problem, so lange man seine Platte partitioniert und die Root-Partition klein hält. (Dies sollte man sowieso immer beachten)

Wenn auf der Platte aber schon ein anderes Betriebssystem installiert ist, kann es unter Umständen unmöglich sein, die BIOS-Einstellungen zu modifizieren, da sonst das alte System nicht mehr lauffähig wäre. Alle neueren Lilo-Distributionen sind jedoch fähig, mit den Einstellungen für LBA- und LARGE-Disk umzugehen.

Die Festplatten, die auschließlich unter Linux benutzt werden und von denen auch nicht gebootet wird, können aus dem BIOS ausgetragen werden. Dadurch bootet das System schneller und Linux erkennt diese Platten später trotzdem automatisch und ohne Zeitverlust. Ich schließe z.B. öfter zusätzlich Festplatten an meinen Rechner an, ohne die BIOS-Einstellung zu verändern.

2.3 Wie beim Booten eingreifen?

Wenn der Lilo-Prompt erscheint, ist es durch Drücken der <Tab> Taste möglich eine Auswahlliste anzuzeigen. Sollte Lilo dafür nicht konfiguriert sein, so hilft das Drücken und Halten der <Alt> oder <Shift> Taste bevor »LILO« angezeigt wird.

Wenn man einen Kernel zum Booten auswählt, kann man nach dem ausgewählten Kernelnamen noch zusätzliche Kommandozeilen-Parameter angeben. Der Kernel akzeptiert umfangreiche Kommandozeilen-Parameter; dies ist aber nicht der richtige Ort, um sie alle aufzulisten. Einige sind aber meiner Meinung nach interessant:

`root=`

diese Partition soll als root gemountet werden, sie kann unterschiedlich sein zu dem Eintrag in `lilo.conf`. Zum Beispiel, habe ich eine kleine Parition mit einem Minimal-Linux, die es mir ermöglicht das System zu booten, sollte durch einen Fehler meine Root-Partition zerstört werden.

`init=`

Version 1.3.43 und neuere Linux-Kernel akzeptieren das Ausführen von anderen Programmen als `/sbin/init`. Wenn beim Booten kritische Probleme auftreten, ist es durch `init=/bin/sh` möglich ein spartanisches System zu starten. Nach Erscheinen des Shell-Prompts ist es in den meisten Fällen nötig, erstmal die Partitionen zu mounten:

```
/sbin/mount -w -n -o remount /; mount -a
```

Vor dem Ausschalten des Computers nicht vergessen,

```
/sbin/umount -a
```

aufzurufen.

eine Nummer

eine Nummer in der Kernel-Kommandozeile veranlaßt *init* den gewünschten *Run-Level* einzunehmen. Der Default ist normalerweise 3. Bitte genaueres in der Dokumentation zu *init*, `/etc/inittab` und `/etc/rc.d` nachlesen.

2.4 Wie Lilo entfernen?

Wenn Lilo einen Boot-Sektor überschreibt, sichert er ihn in `/boot/boot.`*xxyy*. Dabei ist *xxyy* die Major- und Minor-Device-Nummer in Hexadezimal.

```
ls -l /dev/device
```

zeigt einem die Major- und Minor-Nummern an. Zum Beispiel, der erste Sektor von `/dev/hda` (Major 3, Minor 0) wird in `/boot/boot.0300` gespeichert; installiert man Lilo auf `/dev/fd0` wird ein `/boot/boot.0200` angelegt; die Installation von Lilo auf `/dev/sdb3` (Major 8, Minor 19) sichert eine Kopie in `/boot/boot.0819`.

Achtung: Sollte ein solches File schon existieren, wird von Lilo keine Kopie generiert. Darüber muß man sich bei der Installation von Lilo, z.B. nach einer Kernel-Kompilierung, aber nicht sorgen.

Sollte man in die Verlegenheit geraten, Lilo entfernen zu müssen, z.B. bei der De-Installation von Linux, braucht nur der ursprüngliche Boot-Sektor wieder hergestellt werden. Ist Lilo in `/dev/hda` installiert, genügt es,

```
dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1
```

aufzurufen. Ich benutze einfach:

```
cat /boot/boot.0300 > /dev/hda
```

Dies ist aber nicht ganz sicher, da auch die ursprüngliche Partitionstabelle mitrestauriert wird; diese könnte aber in der Zwischenzeit geändert worden sein. Diese Kommando ist aber viel einfacher zu handhaben als ein `fdisk /mbr` in einer DOS shell, denn es erlaubt einem, sauber Linux von der Platte zu entfernen, ohne ein anderes Betriebssystem zu benötigen. Nachdem Entfernen von Lilo sollte man nicht vergessen, mit dem Linux *fdisk* alle Linux-Partitionen zu löschen, da es mit dem DOS *fdisk* nicht möglich ist diese zu löschen.

Ist Lilo auf der Root-Partition installiert, z.B. auf `/dev/hda2`, so ist keinerlei Aktion notwendig, um Lilo zu de-installieren. Nur das Linux *fdisk* aufrufen und alle Linux-Partitionen aus der Paritionstabelle entfernen. Nicht vergessen, die DOS-Partition als Boot-Partition zu markieren.

3 Die einfache Konfiguration

Die meisten Lilo Installationen benutzen eine Konfigurationsdatei wie die folgende:

```
boot = /dev/hda    # oder die Root-Partition, z.B. /dev/hda2
delay = 0          # oder eine Verzögerung in Zehntel-Sekunden
vga = 0            # optional, "vga=1" angeben um 80x50 zu erhalten

image = /boot/vmlinux  # das zImage File (= Kernel)
  root = /dev/hda1     # die Root-Partition
  label = Linux        # oder ein anderer, mehr ausgefallener Name
  read-only            # mounte Root-Partition nur-lesbar

other = /dev/hda4   # die DOS-Partition, so vorhanden
  table = /dev/hda  # die aktuelle Partitionstabelle
  label = dos       # oder ein anderer Name
```

Es sind mehrere `image` und `other` Abschnitte möglich. Es ist nicht unüblich, mehrere Kernel-Images in `lilo.conf` zu konfigurieren; zumindest wenn man mit der Kernel-Entwicklung schritthalten will.

4 Installieren auf `hdc` und Booten von `hda`

Lilo erlaubt es, den Kernel von einer Platte zu mappen und dann das BIOS anzuweisen, ihn von einer anderen Platte zu booten. Z.B. ist es für mich normal, Linux auf einer Platte zu installieren, welche sich unter `hdc` (Master Disk des 2. Kontrollers) befindet, um sie später als Boot-Platte am 1. IDE-Kontroller eines anderen Nur-Linux-Rechners zu benutzen. Ich kopiere dazu die Installations-Diskette auf eine kleine Partition, die ich dann in einer Virtual-Konsole mit *chroot* zur Root-Partition mache. Damit kann ich `hdc` einrichten und während der Installation das System für etwas anderes benutzen.

Die Datei `lilo.conf`,die ich dafür benutze, sieht wie folgt aus:

```
# Dieses File muß von einem Sys. benutzt werden, welches von
# /dev/hdc läuft

boot = /dev/hdc   # überschreibe MBR von hdc
disk = /dev/hdc   # definiere wie hdc dann aussieht:
   bios = 0x80    # das BIOS wird sie dann als erste Platte erkennen

delay = 0
vga = 0

image = /boot/vmlinux  # Pfad des Kernels auf der Root-Partition /dev/hdc1
  root = /dev/hda1     # hdc1 wird beim Booten hda1
  label = Linux
  read-only
```

Dieses Konfigurationsdatei muß von Lilo verarbeitet werden, welcher **von /dev/hdc1** gestartet wurde. Die Lilo Zuordnungstabelle (map), die in den Boot-Sektor `/dev/hdc` geschrieben wird, muß die Dateien `vmlinux` und `/boot/boot.b` des Boot-Zeitpunkts referenzieren, d.h. jene von `hdc`.

Ich nenne diese Konfigurationsdatei `/mnt/etc/lilo.conf.hdc`. Damit installiere ich Lilo durch den Aufruf

```
cd /mnt; chroot .  sbin/lilo -C /etc/lilo.conf.hdc
```

während `/dev/hdc1` unter `/mnt` gemountet ist.

5 BIOS erkennt Root-Partition nicht

Ich habe zwei IDE- und eine SCSI-Platte. Dabei wird die SCSI-Platte nicht vom BIOS erkannt. Der Linux-Lader (LILO) benutzt BIOS-Aufrufe und kann deshalb nur Platten erkennen, die auch das BIOS erkennt. Das dumme AMI BIOS bootet leider nur von »A:« oder »C:«. Mein Root-Dateisystem ist aber auf einer Partition des SCSI-Laufwerks.

Die Lösung ist, den Kernel, die Map-Datei, und den *Chain-Loader* in eine Linux-Partition auf der ersten IDE-Platte zu installieren. Es sei darauf hingewiesen, daß es nicht notwendig ist, den Kernel auf der Root-Partition zu haben.

Die zweite Partition meiner ersten IDE-Platte (`/dev/hda2`, die Linux-Partition zum Booten) ist unter `/u2` gemountet. Hier kommt die `/etc/lilo.conf`, welche ich dafür benutze:

```
#  Installiere LILO in den »Master Boot Record«
#  auf der erste IDE-Platte
boot = /dev/hda

#  /sbin/lilo (der Installierer) kopiert LILOs Boot Record
#  vom folgenden File in den MBR
install = /u2/etc/lilo/boot.b

# Ich schrieb mir ein ausführliches Boot-Menü. LILO findet es unter
message = /u2/etc/lilo/message
```

```
# Der Installierer generiert das folgende File, welches dem
# Boot-Lader sagt wo die Blöcke des Kernels sind.
map = /u2/etc/lilo/map
compact
prompt

#  Warte 10 Sekunden, boote dann den 1.2.1 Kernel als Standard
timeout = 100

#  Der Kernel wird dort installiert wo ihn das BIOS sehen kann, durch:
#       cp -p /usr/src/linux/arch/i386/boot/zImage /u2/z1.2.1
image = /u2/z1.2.1
        label = 1.2.1

#  LILO sagt dem Kernel, dass die erste SCSI-Partition als Root
#  gemountet werden soll. Das BIOS braucht diese Partition nicht
#  zu erkennen.
        root = /dev/sda1

#  Diese Partition wird geprüft und nochmal gemountet durch /etc/rc.d/rc.S
        read-only

#  Ich behalte einen alten Slackware Kernel für den Fall, daß ich einen
#  Kernel generierte habe, welcher nicht funktioniert. (Ich brauchte ihn
#  tatsächlich schon einmal)
image = /u2/z1.0.9
        label = 1.0.9
        root = /dev/sda1
        read-only

#  Meine DR-DOS 6 Partition
other = /dev/hda1
        loader=/u2/etc/lilo/chain.b
        label = dos
        alias = m
```

6 BIOS Probleme mit großen Platten

Das System in meinem Büro hat eine 1GB IDE-Platte. Das BIOS erkennt nur die ersten 504MB dieser IDE-Platte (1MB bedeutet 2**10 bytes, nicht 10**6 bytes). Deswegen habe ich MS-DOS auf einer 350MB Partition `/dev/hda1` und meine Linux Root-Partition auf einer 120MB Partition `/dev/hda2`.

MS-DOS war nicht in der Lage, sich auf der neuen Platte selbst zu installieren. Novell DOS 7 hatte die gleichen Probleme. Glücklicherweise vergaß »Options by IBM« die »OnTrack« Diskette dem Packet mit der Platte beizulegen. Die Platte sollte mit dem »OnTrack Disk Manager« Programm ausgeliefert werden, die für ein Nur-DOS-System, soviel ich weiß, auch benutzt werden muß.

Ich erstellte aber eine Partitionstabelle mit fdisk von Linux. MS-DOS 6.2 verweigerte aber die Installation auf `/dev/hda1`. Die Fehlermeldung lautete ungefähr so: »Diese Version von MS-DOS ist für Neu-Installationen. Ihr Computer hat aber schon ein MS-DOS installiert, deshalb benötigen Sie eine Upgrade-Version von Ihrem Händler.« Dabei war die Platte aber brandneu.

So ein Blödsinn! Ich startete also das Linux fdisk erneut und löschte Partition 1 aus der Tabelle. Das stellte MS-DOS 6.2 zufrieden. Es kreierte anschließend die exakt gleiche Partition 1, welche ich gerade gelöschte hatte und konnte sich darauf installieren. MS-DOS 6.2 schrieb zwar seinen *Master Boot Record* auf die Platte, aber konnte trotzdem nicht starten.

Glücklicherweise hatte ich noch einen Slackware Kernel auf einer Diskette, welche von dem Slackware Installationsprogramm `setup` erstellt wurde. Mit diesem Kernel bootete ich dann Linux und überschrieb den defekten MBR von MS-DOS mit LILO. Das funktionierte endlich. Hier ist die `/etc/lilo.conf` Datei, die ich benutzte:

```
boot = /dev/hda
map = /lilo-map
delay = 100
ramdisk = 0             # schaltet ramdisk im Slackware Kernel aus
timeout = 100
prompt
disk = /dev/hda         # BIOS sieht nur die ersten 500 MB.
   bios = 0x80          # gibt an, daß die erste IDE-Platte benutzt wird
   sectors = 63         # die Geometriedaten aus der HD Dokumentation
   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
```

Danach installierte ich diese Partitionen und überprüfte, daß die Partition mit dem zImage, boot.b, map, chain.b und den message Files, ein MSDOS-Dateisystem benutzen konnte, wenn dieses nicht mit »Stacker« oder »Doublespace« behandelt war. So konnte ich dann DOS von der 500MB Partition `/dev/hda1` booten.

Ich habe auch gelernt, daß »OnTrack« eine um ein paar Bytes versetzte Partitionstabelle geschrieben hätte. Es ist zwar möglich den Linux IDE Treiber zu ändern, damit er diese erkennt, aber die Installation wäre unmöglich mit dem kompilierten Slackware Kernel. Schließlich erhielt ich doch noch die »OnTrack« Diskette. Ich rief OnTracks Technischen Support an, welcher mir erzählte, daß Linux fehlerhaft sei, da es nicht das BIOS benutzt. Ich habe ihre Diskette dann verschenkt.

7 Booten von einer Notfall-Diskette

Als nächstes installierte ich Windows95 auf meinem Büro-Rechner. Es zerstörte meinen LILO-MBR, aber ließ meine Linux-Partitionen unverändert. Da es lange dauert einen Kernel von der Diskette zu booten, installierte ich mir LILO auf einer Diskette, um damit den Kernel von IDE zu booten.

Ich erstellte die LILO-Diskette wie folgt:

```
fdformat /dev/fd0H1440      #  formatiere eine neue Diskette
mkfs -t minix /dev/fd0 1440 #  erstelle ein minix Dateisystem
mount /dev/fd0 /mnt         #  mounte Disk unter Standard Mount-Verzeichnis
cp -p /boot/chain.b /mnt    #  Kopiere den chain loader
lilo -C /etc/lilo.flop      #  installiere LILO und map auf Diskette
umount /mnt
```

Bitte beachten: die Diskette **muß gemountet sein, wenn der Installierer gestartet wird**, damit Lilo sein map-File richtig schreiben kann.

Diese Datei nannte ich `/etc/lilo.flop` und sie stimmt mit der letzten fast überein:

```
# Erstelle Diskette, die den Kernel von HD bootet

boot = /dev/fd0
map = /mnt/lilo-map
delay = 100
ramdisk = 0
timeout = 100
prompt
disk = /dev/hda     # 1 GB IDE, BIOS sieht nur die ersten 500MB
   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
```

Schließlich brauchte ich noch MS-DOS 6.2 auf meinem Büro-Rechner; allerdings wollte ich nicht die erste Platte anrühren. Ich kaufte mir einen SCSI-Controller und eine SCSI-Platte, erstellte darauf mit dem Linux mkdosfs ein MS-DOS Dateisystem, das Windows-95 als »D:« sieht. Natürlich bootet MSDOS nicht von »D:«, das ist aber mit LILO kein Problem. Ich fügte folgendes zu der `lilo.conf` aus Beispiel 2:

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

Mit dieser Ergänzung läuft MS-DOS 6.2. MS-DOS glaubt, es befindet sich auf »C:« und Windows-95 ist auf »D:«.