Kapitel 8

Systemstart

In diesem Kapitel geht es um den Start eines Linux-Systems. Der Systemstart wird primär durch zwei Komponenten bestimmt:

- **LILO** ist dafür verantwortlich, dass der Linux-Kernel gestartet wird. Dazu wird LILO (ein winziges Programm) auf eine Diskette oder in den Bootsektor einer Festplatte installiert. Diese Installation kann je nach Hard- und Software recht trickreich sein und wird in allen Facetten beschrieben. Behandelt werden unter anderem die Erstellung von Bootdisketten, die LILO-Deinstallation, das 1024-Zylinder-Limit etc.
- Der **Init-V-Prozess** (das Programm `init`) kümmert sich darum, dass unmittelbar nach dem Kernel verschiedene Initialisierungsarbeiten durchgeführt werden (Dateisysteme einbinden etc.) und dass diverse Systemprozesse gestartet werden (beispielsweise für alle Netzwerkfunktionen). Ein Grundverständnis dieses Prozesses hilft sehr bei der Durchführung von Konfigurationsarbeiten.

Neben diesen Schwerpunkten werden einige weitere Themen eher knapp angerissen:

- LILO-Alternativen (GRUB, Loadlin, SYSLINUX etc.)
- Kernel-Bootoptionen zur Umgehung von Hardware-Problemen beim Rechnerstart
- Logging-Dateien als Hilfsmittel bei der Fehlersuche

VERWEIS

Die Informationen in diesem Kapitel sollten auch dann weiterhelfen, wenn Sie Ihren Rechner nicht mehr starten können. Weitere Erste-Hilfe-Tipps finden Sie auf Seite 102 sowie im Anhang, wo für mehrere Distributionen der Umgang mit Notfall-Disketten und -CDs kurz beschrieben wird.

Nach LILO und `init` wird üblicherweise auch X gestartet. Details zum X-Startprozess finden Sie ab Seite 526.

8.1 LILO

Zum Booten ist es erforderlich, dass die Linux-Kernel-Datei von der Diskette oder der Festplatte ins RAM geladen und dann gestartet wird. Der Kernel muss alle Treiber zum Zugriff auf die Root-Partition enthalten. Das Laden und Starten des Kernels ist die Aufgabe von LILO (Linux Loader). Dieses Programm steht daher im Mittelpunkt dieses Kapitels.

LILO startet nicht nur Linux, es kann auch dazu verwendet werden, andere Betriebssysteme (DOS, OS/2, diverse Windows-Versionen) hochzufahren. LILO ist insofern ein Schlüsselelement im Hinblick auf Dual- oder Multi-Bootsysteme, bei denen Sie nach dem Rechnerstart angeben können, welches Betriebssystem geladen werden soll.

Dieser Abschnitt hat immer die friedliche Koexistenz mehrerer Betriebssysteme im Auge. Falls auf Ihrem Rechner als einziges Betriebssystem Linux installiert ist, brauchen Sie keine Rücksicht auf andere Betriebssysteme zu nehmen. In diesem Fall ist die LILO-Konfiguration weitgehend trivial, und Sie können den Großteil dieses Abschnitts einfach überblättern. (Es ist aber dennoch sinnvoll, dass Sie die Funktionsweise von LILO verstehen. Außerdem müssen Sie auch bei einem reinen Linux-System auf das 1024-Zylinder-Limit Rücksicht nehmen.)

VORSICHT

Die LILO-Installation zählt neben der Partitionierung der Festplatte zu den kritischsten Teilen einer Linux-Installation. Wenn Sie dabei blind dem Installationsprogramm Ihrer Distribution vertrauen, ist die Chance recht groß, dass alles auf Anhieb funktioniert. Leider kann aber auch einiges schief gehen – und mit etwas Pech können Sie danach weder Linux noch ihr altes Betriebssystem booten!

Dieses Dilemma können Sie vermeiden, wenn Sie den folgenden Ratschlag befolgen: Erstellen Sie während der Linux-Installation eine Bootdiskette bzw. installieren Sie LILO auf eine Diskette (nicht auf die Festplatte)! Fast alle Distributionen bieten diese Möglichkeit.

Mit dieser Diskette können Sie Linux booten. Dann sollten Sie sich die Mühe machen und diesen Abschnitt lesen, bevor Sie LILO auf Ihre Festplatte installieren. Lesen Sie insbesondere die Abschnitte ab Seite 329, wo es darum geht, ein Backup des Bootsektors zu erstellen bzw. den Bootsektor nach einer fehlgeschlagenen LILO-Installation wieder herzustellen!

Überblick

Bevor es mit den zahlreichen LILO-Details losgeht, hilft ein kurzer Überblick bei der Orientierung in diesem recht langen Abschnitt.

- **LILO bedienen:** Vielleicht haben Sie Linux (samt LILO) schon installiert? Nach dem Rechnerstart (egal, ob von einer Diskette oder von der Festplatte) erscheinen die vier Buchstaben 'lilo' auf dem Bildschirm. Seite 318 beschreibt, wie es dann weitergeht.

- **LILO-Grundlagen:** Ganz egal, ob Sie LILO manuell einrichten (Konfigurationsdatei `/etc/lilo.conf`) oder ob Sie dazu ein Tool Ihrer Lieblingsdistribution einsetzen – Sie sollten wissen, was Sie dabei eigentlich tun! Ab Seite 319 wird das erforderliche Grundlagenwissen vermittelt und ab Seite 321 das Format der LILO-Konfigurationsdatei beschrieben.
- **LILO-Bootdisketten:** LILO kann wahlweise auf die Festplatte oder auf eine Diskette installiert werden. Erste Experimente sollten unbedingt mit einer Diskette durchgeführt werden! Ab Seite 326 werden drei verschiedene Möglichkeiten beschrieben, Bootdisketten zu erzeugen.
- **LILO-Installation auf der Festplatte:** Noch eleganter ist natürlich die LILO-Installation auf die Festplatte. Beim Rechnerstart können Sie zwischen verschiedenen Betriebssystemen auswählen (z. B. Windows oder Linux) – und das ohne langsame Zugriffe auf Disketten. Allerdings besteht die Gefahr, dass Sie nach einer fehlerhaften LILO-Installation auf die Festplatte weder Windows noch Linux starten können! Daher ist bei diesem Schritt Vorsicht angebracht. Auf Seite 329 wird nicht nur die Installation von LILO in den MBR (Master Boot Record) beschrieben, sondern auch, wie Sie vorher eine Sicherheitskopie dieses Sektors erstellen und wie Sie LILO später einmal deinstallieren können.
- **LILO und das 1024-Zylinder-Limit:** Im Gegensatz zu Linux ist LILO auf das BIOS angewiesen. Dieses ist allerdings heute noch kompatibel mit Systemen aus der Computer-Steinzeit und verursacht daher häufig Probleme: Unter bestimmten Umständen können Sie Linux nur starten, wenn sich die Bootdateien in einer Partition am Anfang der Festplatte befinden. Ab Seite 330 finden Sie Hintergrundinformationen zu diesem Limit – und Tipps, wie dieses Limit umgangen werden kann.
- **LILO und Windows NT/2000/XP:** LILO ist leider nur unter bestimmten Voraussetzungen in der Lage, Windows NT/2000/XP zu starten. Dennoch stellt es kein Problem dar, ein Multibootsystem für Linux und Windows NT/2000/XP einzurichten. Ab Seite 336 wird beschrieben, wie das geht.
- **SCSI-, RAID, LVM-, Reiserfs-Systeme starten:** Damit der Linux-Kernel nach seinem Start auf die Systempartition (Root-Partition) zugreifen kann, benötigt er gegebenenfalls SCSI-, RAID-, LVM- oder spezielle Dateisystemtreiber. Wenn diese nicht Bestandteil des Kernels sind, müssen sie als Module von einer RAM-Disk geladen werden. Details dazu finden Sie ab Seite 332.
- **LILO-Fehlermeldungen:** Wenn der Start von Linux durch LILO nicht klappt, versucht LILO zumindest Fehlermeldungen anzugeben. Deren Interpretation ist allerdings eine eigene Kunst. Ab Seite 339 gibt es eine Einführung in diese Kunst.
- **LILO-Reboot:** Wenn Sie den Rechner für einen Neustart herunterfahren, können Sie mit `lilo -R` angeben, wie der Rechner anschließend neu gestartet werden soll.
- **Menüs und Grafikmodus:** Mit den aktuellen LILO-Versionen können Sie die Auswahl des gewünschten Betriebssystems mit einem Menü vereinfachen bzw. mit einer Hintergrundgrafik ansprechender gestalten – siehe Seite 341.
- **LILO-Konfigurationshilfen:** Sie können sich bei der LILO-Konfiguration von den Tools der diversen Distributionen helfen lassen. Das spart etwas Tippaufwand, führt

aber meist nur dann zum gewünschten Ergebnis, wenn Sie auch verstehen, was Sie tun (und wenn das Tool der jeweiligen Distribution die Gegebenheiten Ihres Systems richtig erkennt, was leider nicht immer der Fall ist). Seite 342 zählt einige derartige Programme auf.

VERWEIS

Zu LILO gibt es eine umfangreiche Online-Dokumentation. `man lilo.conf` und `man lilo` beschreiben das Installationsprogramm und das Format der Steuerungsdatei. Eine noch detailliertere Beschreibung wird bei den meisten Distributionen als Teil des LILO-Pakets mitgeliefert (siehe `rpm -qd lilo`, Dateien `text.ps` und `user.ps`). Lesenswert sind auch die diversen (Mini-)HOWTOs: Bootdisk, Boot+Root+Raid+LILO, LILO, LILO-crash-rescue, Linux+WinNT sowie Multiboot-with-LILO.

Einige Alternativen zu LILO werden in einem eigenen Abschnitt ab Seite 343 vorgestellt.

Zu guter Letzt enthalten die Anhänge dieses Buchs einige distributionsspezifische Tipps zu den Themen Bootdisketten, Rescue-System etc.

Bedienung von LILO

Dieser Abschnitt setzt voraus, dass LILO bereits auf einer Diskette oder auf der Festplatte installiert ist. Je nach Installation erscheint LILO unmittelbar nach dem Rechnerstart in Form einer schönen Grafik mit Menü-Bedienung per Cursor-Tasten, oder aber ganz spartanisch im Textmodus durch den Text 'LILO boot:'.

Normalerweise ist LILO so eingestellt, dass das Programm einige Sekunden darauf wartet, ob der Anwender irgendwelche Eingaben durchführen möchte. Ist das nicht der Fall, wird automatisch das bei der Konfiguration eingestellte Defaultsystem gestartet (zumeist Linux). Die Wartezeit können Sie verkürzen, indem Sie einfach (↵) drücken.

Möglicherweise möchten Sie in den Bootvorgang eingreifen, etwa um ein anderes Betriebssystem auszuwählen oder um zusätzliche Bootoptionen zu übergeben (siehe auch Seite 356). In diesem Fall hängt die Bedienung vom LILO-Erscheinungsmodus ab.

Textmodus: In sehr seltenen Fällen müssen Sie als Erstes (Shift) drücken, um in den interaktiven Modus zu gelangen. Anschließend zeigt LILO eine Liste mit den Namen der zur Auswahl stehenden Betriebssysteme an, sobald Sie die Taste (Tab) drücken. Zur Auswahl stehen meist `linux` und `windows`. LILO kann aber auch so konfiguriert werden, dass Sie die Wahl zwischen mehreren Linux-Distributionen oder zwischen unterschiedlichen Kernel-Versionen haben. SuSE-Linux sieht sogar den Start eines Programms zum Speichertest (RAM) vor.

Auf jeden Fall können Sie nun zuerst einen der zur Auswahl stehenden Namen und dann die gewünschten zusätzlichen Parameter eintippen. Mit dem folgenden Kommando wird Linux gestartet. Allerdings wird statt der vordefinierten Defaultpartition `/dev/hdc7` als Root-Partition verwendet:

```
LILO
boot:  <Tab>
linux    windows   memory
boot:  linux root=/dev/hda7
```

Grafikmodus: Bei manchen LILO-Installationen beendet (Esc) den Grafikmodus und Sie gelangen in den Textmodus – siehe oben. Ansonsten können Sie mit den Cursor-Tasten eines der Betriebssysteme auswählen und dann über die Tastatur zusätzliche Parameter eingeben (ohne den Grafikmodus zu verlassen).

HINWEIS

Beachten Sie, dass bei der LILO-Eingabe normalerweise das amerikanische Tastaturlayout gilt! (Y) und (Z) sind vertauscht, die Sonderzeichen befinden sich an ungewohnten Stellen. Eine Übersetzungstabelle für die deutsche Tastatur finden Sie auf Seite 97. (Seit LILO-Version 20 ist es zwar möglich, ein anderes Tastaturlayout einzustellen, diese Option wird aber leider selten genutzt.)

LILO-Interna

Der LILO (Linux Loader) ist ein winziges Programm, das im Bootsektor (also im ersten Sektor) einer Diskette, Festplatte oder Festplattenpartition installiert werden kann. Das Programm ermöglicht die Auswahl zwischen mehreren installierten Betriebssystemen und insbesondere den Start von Linux. LILO ist die schnellste und beliebteste Methode, um Linux zu starten. (Dieses Kapitel beschreibt die LILO-Version 21.7.)

VORSICHT

Die LILO-Installation ist eine kritische Angelegenheit, besonders dann, wenn LILO in den MBR (Master Boot Record) der Festplatte installiert wird. Sollte dabei etwas schief gehen, können Sie Ihren Rechner ohne Bootdiskette weder unter Ihrem bisherigen Betriebssystem noch unter Linux hochfahren! Besonders problematisch ist die LILO-Installation auf Rechnern mit Windows NT/2000/XP, wenn die erste Festplattenpartition eine NTFS-Partition ist (d. h. keine FAT-Partition). Wenn Sie bei einer derartigen Konstellation LILO in den MBR installieren, können Sie anschließend möglicherweise Windows nicht mehr starten!

VORSICHT

Auch wenn Sie den Disk-Manager EZ-Drive verwenden, was wahrscheinlich nur auf sehr wenigen sehr alten Rechnern der Fall ist, dürfen Sie LILO auf keinen Fall im MBR installieren! Diese Einschränkung gilt vermutlich auch für andere Disk-Manager. (Disk-Manager erlauben den Zugriff auf große IDE-Platten trotz einer sehr alten BIOS-Version, siehe Seite 98.) Wenn Sie LILO dennoch verwenden möchten, ist nur eine Installation auf einer Diskette oder im ersten Sektor der Linux-Partition möglich – siehe die LILO-Dokumentation!

Bei den meisten Distributionen kann die LILO-Konfiguration als Teil des Installationsprozesses erfolgen. Wegen der oben beschriebenen Risiken hat eine manuelle LILO-Installation aber Vorteile. Sie haben eine viel bessere Kontrolle darüber, was wirklich passiert, können eine Backup-Diskette für den MBR erstellen etc. LILO in den MBR zu

installieren ist keine Voraussetzung für den Betrieb von Linux! In den ersten Tagen – bis Sie ein wenig Sicherheit im Umgang mit Linux gewonnen haben – können Sie ohne weiteres auch von einer Diskette booten.

VORSICHT

Noch zwei Warnungen: Wenn Sie zuerst LILO und später irgendeine Version von Microsoft Windows installieren, wird LILO ungefragt überschrieben. An sich ist es kein Problem, LILO anschließend wieder herzustellen. Allerdings benötigen Sie eine Bootdiskette, damit Sie Linux überhaupt starten können (um dann dort das Kommando `lilo` auszuführen). Werfen Sie Ihre Bootdiskette also auch nach einer erfolgreichen LILO-Installation nicht weg!

Wenn Sie einen neuen Kernel installieren (oder den Kernel neu kompilieren), müssen Sie LILO neu installieren (siehe Seite 398). Denken Sie daran, auch Ihre Bootdiskette zu aktualisieren!

Funktionsweise

Die Funktion von LILO besteht (vereinfacht ausgedrückt) darin, dass er die Datei mit dem Linux-Kernel lädt und ausführt. Das hört sich allerdings einfacher an, als es in Wirklichkeit ist:

Genau genommen befindet sich im Bootsektor der Festplatte bzw. der Diskette ein winziges Programm (*first stage*), dessen einzige Aufgabe darin besteht, das größere LILO-Hauptprogramm zu laden (*second stage*). (Der Bootsektor, also der erste Sektor der Festplatte oder Diskette, wäre zu klein, um das gesamte LILO-Programm zu speichern! Der Code im Bootsektor wird vom BIOS beim Rechnerstart automatisch ausgeführt – daher spielt der Bootsektor eine derart wichtige Rolle für den Rechnerstart.)

Das LILO-Hauptprogramm (es handelt sich um die Datei `/boot/boot.b`) befindet sich irgendwo auf der Festplatte. Der Bootsektor mit dem Startprogramm enthält unter anderem die Nummern der Sektoren, auf denen das Hauptprogramm zu finden ist.

Sobald das LILO-Hauptprogramm läuft, bietet es dem Anwender die Möglichkeit, via Tastatur zwischen verschiedenen Betriebssystemen auszuwählen (siehe Seite 318). Wenn sich der Anwender für Linux entscheidet, muss LILO die Kernel-Datei laden. Dazu wurde bei der Installation von LILO eine Tabelle erstellt (die Datei `/boot/map`), die der Reihe nach alle Sektoren enthält, auf denen sich die Kernel-Datei befindet.

Der Grund für diese ungewöhnliche Vorgehensweise ist folgender: Während LILO läuft, fehlen noch jegliche Informationen über DOS-, Linux- oder andere Dateisysteme. Die Festplatte ist aus der Sicht von LILO eine riesige Ansammlung von Datensektoren. LILO muss daher in jeder Phase wissen, wo sich die als Nächstes benötigten Datensektoren befinden.

Aufgrund der obigen Beschreibung sollte auch klar sein, dass LILO nur funktioniert, solange der Ort der Kernel-Datei auf der Festplatte unverändert bleibt. (Jedes Kopierkommando und insbesondere jedes Neukompilieren des Kernels löst eine Veränderung aus, auch wenn der Dateiname unverändert bleibt! LILO muss in solchen Fällen durch die

Ausführung des Kommandos `lilo` neu installiert werden! `lilo` führt dabei die Vorbereitungsarbeiten durch, d. h. es erzeugt aktualisierte Versionen von `/boot/boot.b` und `/boot/map` und speichert deren Startsektoren und andere Informationen im Bootsektor.)

Einschränkungen

LILO wird ausgeführt, bevor irgendein Betriebssystem läuft. Aus diesem Grund ist LILO – im Gegensatz zu Linux – auf das BIOS angewiesen, um Sektoren von der Festplatte zu lesen. Ältere BIOS-Versionen lassen aus historischen Gründen aber nur einen Zugriff auf die ersten 1024 Zylinder der Festplatte zu. Die sich daraus ergebenden Probleme (samt Lösungsvorschlägen) sind ab Seite 330 beschrieben. Lesen Sie unbedingt diesen Abschnitt, wenn Sie ein altes Mainboard verwenden (alt bedeutet vor ca. 1998).

Neben dem Zylinder-Limit existiert eine zweite Einschränkung: Ältere BIOS-Versionen können während des Bootprozesses nur die beiden ersten Festplatten ansprechen. Auf einem System mit mehreren Festplatten muss sich die Kernel-Datei also auf einer der beiden ersten Platten befinden. (Auch diese Einschränkung gilt nur bei alten Mainboards. Moderne Mainboards bzw. SCSI-Controller kennen dieses Problem nicht.)

Sowohl das Zylinder-Limit als auch die Limitierung auf zwei Festplatten gelten (wenn überhaupt) nur für LILO bzw. für den Ort der LILO-Dateien (inklusive Kernel). Sobald der Kernel einmal läuft, können alle Festplatten vollständig genutzt werden. Die `root`-Partition) von Linux kann sich also durchaus auf der fünften Platte ab Zylinder 2500 befinden, solange nur der Kernel für LILO erreichbar ist.

Komplikationen kann es schließlich geben, wenn Sie unter Linux statt `ext2` ein anderes Dateisystem verwenden, wenn Sie mit SCSI- statt mit IDE-Festplatten arbeiten oder wenn Sie RAID oder LVM einsetzen. Die Probleme können zwei Ursachen haben: Erstens kann es passieren, dass LILO die Sektoren der Kernel-Datei nicht zuordnen kann. (Dieses Problem tritt bereits bei der Installation von LILO auf.) Zweitens kann es passieren, dass LILO zwar den Kernel erfolgreich laden und starten kann, dieser dann aber nicht in der Lage ist, das Root-Dateisystem zu lesen. Hintergrundinformationen und Lösungsvorschläge finden Sie auf Seite 332.

LILO kann momentan außer Linux die Betriebssysteme DOS, Windows 3.1/9x/ME (entspricht im Bootprozess jeweils DOS), OS/2 und einige Unix-Varianten booten. Nicht unterstützt wird leider Windows NT/2000/XP, das auf einen eigenen Bootloader angewiesen ist. Bei Windows NT/2000/XP kann aber umgekehrt der NT-Bootloader dazu eingesetzt werden, um LILO zu starten. Dieser Kooperationsweg ist ab Seite 336 beschrieben.

LILO-Konfiguration

Die Installation des LILO besteht aus zwei Schritten: Zuerst wird die Konfigurationsdatei `/etc/lilo.conf` erstellt, anschließend das Kommando `lilo` ausgeführt. Dieses Kommando wertet die Konfigurationsdatei aus, erstellt daraus einen neuen Bootsektor und

schreibt diesen an den durch `lilo.conf` angegebenen Ort. Dabei handelt es sich zumeist um den MBR einer Festplatte oder Diskette, es kann sich aber auch um eine gewöhnliche Datei handeln.

TIPP

Selbst wenn Sie `lilo.conf` nicht ändern, müssen Sie `lilo` jedes Mal aufrufen, wenn sich die Kernel-Datei verändert (z. B. nach einem neuen Kompilieren)! Für LILO ist nicht der Dateiname relevant, sondern die Sektoren, in denen die Datei gespeichert ist.

Die Datei `lilo.conf` besteht aus zwei Teilen: Der erste Teil steuert das generelle Verhalten des Bootprogramms, der zweite Teil (Schlüsselwort `image` bzw. `other`) listet alle Betriebssysteme auf, die durch LILO gestartet werden (DOS, Windows, Linux). Das erste Betriebssystem dieser Liste gilt automatisch als Defaultbetriebssystem. Kommentare werden durch das Doppelkreuz (#) eingeleitet.

Im Prinzip können Sie in jeder Festplattenpartition ein eigenes Betriebssystem (etwa unterschiedliche Linux-Distributionen) installieren und LILO zur Auswahl der gewünschten Partition verwenden. Sie können LILO auch dazu verwenden, um zwischen verschiedenen Linux-Kernel-Dateien innerhalb einer Partition zu unterscheiden (etwa `vmlinuz`, `vmlinuz.old`). Das ist insbesondere dann interessant, wenn Sie eine neue Linux-Version testen möchten, ohne auf die alte zu verzichten.

Globale LILO-Optionen

boot: Der erste Teil von `/etc/lilo.conf` beginnt mit der Anweisung `boot=` und gibt an, wohin LILO installiert werden soll. Zur Installation auf eine Diskette geben Sie `/dev/fd0` an. Um LILO in den MBR der ersten IDE-Platte zu installieren, verwenden Sie `/dev/hda`, für die erste SCSI-Platte `/dev/sda`.

prompt: `prompt` bewirkt, dass LILO den Prompt (die Zeichen 'boot:') anzeigt und so verdeutlicht, dass jetzt Eingaben möglich sind. `prompt` sollte immer verwendet werden. Wird `prompt` nicht verwendet, erscheint der Eingabe-Prompt nur, wenn die Shift-Taste gedrückt wird.

delay: `delay` gibt an, wie viele Zehntelsekunden `LILO` beim Booten auf ein manuelles Eingreifen wartet. Am schnellsten geht es mit 0, dann müssen Sie aber schon vor dem Start von LILO Shift gedrückt halten, wenn Sie nicht das Defaultbetriebssystem verwenden möchten.

Wenn Sie nur `prompt`, nicht aber `delay` angeben, wartet Linux unbegrenzt auf die Auswahl eines Betriebssystems. Damit ist ein unbeaufsichtigter Neustart ausgeschlossen. Wenn Linux als (Netzwerk-)Server eingesetzt wird, ist das aber nicht sinnvoll! Hier sollte LILO so konfiguriert werden, dass Linux (z. B. nach einem Stromausfall) ungefragt und ohne weitere Eingriffe sofort wieder gestartet wird.

lba32: `lba32` umgeht das 1024-Zylinder-Limit. `lilo` speichert Sektornummern nicht als CHS-Tripel (Cylinder, Head, Sector), sondern im 32-Bit-LBA-Format (Logical Block Addressing). Generell sollten Sie diese Option immer verwenden, wenn sich die Kernel-

Dateien auf einer großen Festplatte befinden und Sie ein modernes Mainboard besitzen (ca. ab 1998). Die Hintergründe des 1024-Zylinder-Limit sind auf Seite 330 beschrieben.

linear: Die Option `linear` ist eine Alternative zu `lba32`. Sie bewirkt, dass die Sektoradressen in `/boot/map` als herkömmliche LBA-Werte gespeichert werden. Wenn sich LILO und das BIOS bei der Interpretation der CHS-Geometrie der Festplatte nicht einig sind, bietet `linear` oft die einfachste Lösung. `linear` muss oft verwendet werden, wenn Sie ein altes Mainboard oder einen alten SCSI-Controller verwenden. Sie können auf diese Weise nur die ersten 1024 Zylinder der Festplatte ansprechen.

compact: `compact` ermöglicht ein besonders schnelles Laden des Kernels (besonders von der Diskette), funktioniert aber nicht auf jedem Rechner bzw. mit jedem Festplatten-Controller.

bootmap: Mit der Option `map` kann die Datei angegeben werden, in der die Sektornummern der Kernel-Dateien und anderer Dateien gespeichert werden. Ohne diese Option wird `/boot/map` verwendet, was zumeist eine sinnvolle Voreinstellung ist. Die Option muss nur verwendet werden, wenn sich die Datei an einem anderen Ort befindet (z. B. auf einer Bootdiskette).

install: Eine weitere Option, auf deren Angabe oft verzichtet wird, ist `install`: Damit wird angegeben, in welche Datei das LILO-Hauptprogramm (der second stage loader) gespeichert wird. Die Defaulteinstellung lautet `/boot/boot.b`.

```
# Beispiel für /etc/lilo.conf (Teil 1)
# LILO global section
boot = /dev/fd0        # Installation im MBR einer Diskette
delay = 100            # 10 Sekunden warten
# prompt               # Eingabe erzwingen (kein automatischer Start)
# compact              # schneller, besonders bei Disketten; kann bei
                       # manchen Festplatten Probleme verursachen
# map=/boot/map        # ohnedies Defaulteinstellung
# install=/boot/boot.b # ohnedies Defaulteinstellung
# linear               # manchmal bei alter Hardware erforderlich
lba32                  # nicht bei alten Mainboards (vor 1998)!
```

In manchen Fällen (große Platten und altes BIOS) hat LILO Probleme mit der Festplattengeometrie. Als ersten Lösungsversuch sollten Sie es mit der `linear`-Option versuchen. Wenn auch das nicht hilft, müssen Sie die Option `disk` mit deren Suboptionen `bios`, `sectors`, `heads` und `cylinders` verwenden. Hintergrundinformationen zu diesem Thema finden Sie auf Seite 98 sowie in der LILO-Dokumentation.

```
disk=/dev/hda          # Zusatzinformationen für Device /dev/hda
  bios=0x80            # 0x80 für die 1. Platte,
                       # 0x81 für die 2. Platte ...
  sectors=63           # Anzahl der Sektoren
  heads=255            # Anzahl der Köpfe
  cylinder=522         # Anzahl der Zylinder
```

LILO-Optionen für den Start von Linux

Im zweiten Teil von `lilo.conf` sind der Reihe nach bis zu 16 Betriebssystemvarianten aufgezählt, die wahlweise gestartet werden können. Die erste Variante gilt als Defaulteinstellung. Gemeinsames Merkmal aller Einträge ist das Kommando `label`, das der jeweiligen Variante einen Namen gibt. Diese Namen sind bei der manuellen Auswahl des Betriebssystems einzugeben – wählen Sie also kurze und aussagekräftige Namen ohne Leer- und Sonderzeichen.

TIPP

Sofern bei der LILO-Konfiguration keine spezielle Tastaturtabelle angegeben wird (Schlüsselwort `keytable`, siehe LILO-Dokumentation), gilt für die Eingabe der Label-Namen beim LILO-Start das US-Tastaturlayout. Wenn Sie mit einer deutschen Tastatur arbeiten, vermeiden Sie Label-Namen mit Sonderzeichen und mit den Buchstaben Y und Z!

Um Linux zu booten, müssen außer `label` noch die Kommandos `image` und `root` angegeben werden. `image` bestimmt den Ort der Kernel-Datei, `root` die Partition, auf der sich das Wurzelverzeichnis befindet.

read-only: `read-only` gibt an, dass die Root-Partition zuerst read-only gemountet wird. Der Init-Prozess kann dann das Dateisystem kontrollieren und gegebenenfalls reparieren, bevor es im Read-Write-Modus neu (und endgültig) gemountet wird. Diese Option sollte immer verwendet werden!

append: Mit `append` können zusätzliche Kernel-Optionen angegeben werden (etwa um Hardware-Probleme zu vermeiden). Die wichtigsten Optionen sind auf Seite 356 beschrieben. Die Optionen gelten nur für in den Kernel integrierte Funktionen, nicht aber für Module, die später geladen werden. (Moduloptionen werden in `/etc/modules.conf` angegeben – siehe Seite 377.)

intird: `initrd` gibt den Namen einer RAM-Disk-Datei an, die von LILO geladen werden soll. Das bietet die Möglichkeit, gleich nach dem Start des Kernels ein Installationssystem zu starten oder Module nachzuladen. Das ist wichtig, wenn Sie SCSI-Festplatten oder ein besonderes Dateisystem verwenden (siehe Seite 332).

vga: `vga` bestimmt den VGA-Modus: Mögliche Einstellungen sind `extended` für einen Textmodus mit 50 Zeilen, `normal` für den Standardtextmodus und eine beliebige Zahl n größer 0 für den gewünschten VGA-Modus n. (Vorsicht! Wenn Sie einen Modus angeben, den Ihre VGA-Karte nicht kennt, können Sie Linux nicht korrekt verwenden!)

```
# Beispiel für /etc/lilo.conf (Teil 2)
# Linux images
image = /boot/vmlinuz                # Kernel-Datei
        label = linux                # Name
        root = /dev/hda8             # Root-Device
        read-only                    # zuerst read-only laden
        # initrd = /boot/initrd      # SCSI/RAID/LVM/reiserfs etc.
```

```
# Alternative: 'linuxbak' zum Booten einer älteren Kernel-Version
image= /boot/vmlinuz.bak              # Kernel-Datei
        label = linuxbak              # Name
        root = /dev/hda8              # Root-Device
        read-only                     # zuerst read-only laden
        # initrd = /boot/initrd       # SCSI/RAID/LVM/reiserfs etc.
        # append = "aic7xxx=extended" # Kernel-Option für SCSI-Karte
```

LILO-Optionen für den Start von Windows

Bei DOS- und Windows-3.1/9x/ME-Varianten wird das Schlüsselwort `other` zur Angabe der Partition verwendet. Mit den folgenden Zeilen wird das Betriebssystem gestartet, das sich in der ersten Partition der ersten Festplatte befindet.

```
# Datei /etc/lilo.conf, Teil 3
# DOS/Windows 3.1/9x/ME
other = /dev/hda1                     # DOS/Windows-Partition
        label = windows               # Name
```

HINWEIS

Die obigen Zeilen starten unter Umständen auch den Boot Loader von Windows NT/2000/XP – aber leider nicht immer. Der Start funktioniert bei manchen Windows-Installationen, wenn die erste Partition der Festplatte eine FAT-Partition ist (nicht NTFS!) und der NT-Boot-Loader im Bootsektor dieser Partition untergebracht ist.

Diese Voraussetzungen sind nicht bei allen Windows-Installationen gegeben. Es kann sein, dass der NT/2000/XP-Boot-Loader im MBR der Festplatte untergebracht ist bzw. dass der NT-Boot-Loader bestimmte Daten im MBR erwartet, die er dort (nach einer LILO-Installation) nicht findet. Daher sollten Sie bei Windows NT/2000/XP LILO nie in den MBR der Festplatte installieren und stattdessen den auf Seite 336 beschriebenen Weg einschlagen.

Die beiden folgenden Zeilen sehen fast so aus wie oben – aber es gibt einen wesentlichen Unterschied. Hier wird nicht eine bestimmte Partition angegeben, sondern die gesamte Festplatte (keine Partitionsnummer!). Damit ist der MBR der Festplatte gemeint, der von LILO ausgeführt wird. Das ist nur sinnvoll, wenn nicht LILO selbst in eben diesen MBR installiert wurde (sondern in den MBR einer anderen Festplatte oder einer Diskette). Eine Anwendung dieser Variante ist nur sinnvoll, wenn LILO auf eine Diskette installiert wird oder wenn der Rechner mit mehreren bootfähigen Festplatten ausgestattet ist.

```
other = /dev/hdb                      # MBR der 2. Festplatte
        label = windows               # Name
```

HINWEIS

Eine Menge weiterer LILO-Optionen werden in der Online-Dokumentation beschrieben – etwa `message` zur Ausgabe eines Informationstextes, `keytable` zur Definition eines bestimmten Tastaturlayouts, `password` für den Passwortschutz des Bootprozesses etc.

LILO-Bootdiskette

Selbst wenn Sie längerfristig LILO auf die Festplatte installieren möchten, benötigen Sie zuerst eine Bootdiskette! Der Grund: Wenn bei der LILO-Installation etwas schief geht, wenn bei einer späteren Installation von Windows LILO überschrieben wird etc., benötigen Sie eine Möglichkeit, Linux dennoch starten zu können und gegebenenfalls LILO neu einzurichten.

Im Regelfall haben Sie bereits während der Linux-Installation die Möglichkeit, eine Bootdiskette zu erstellen. Außerdem werden mit den meisten Distributionen Werkzeuge mitgeliefert, um komfortabel neue Bootdisketten zu erstellen. Im Anhang sind derartige Programme für einige Distributionen beschrieben.

Wenn Sie selbst Hand anlegen möchten, gibt es mehrere Möglichkeiten, Linux-Bootdisketten zu erzeugen:

- LILO-Bootdisketten ohne Kernel: Hier wird LILO in den MBR der Diskette geschrieben. LILO greift dann auf die Kernel-Datei der Festplatte zu (d. h. der Kernel befindet sich nicht auf der Diskette, daher die Bezeichnung 'ohne Kernel'). Vorteil: Sehr schneller Bootprozess (von der Diskette werden nur wenige Byte gelesen). Nachteile: Wenn der Kernel auf der Festplatte nicht gefunden wird (nach einem Neukompilieren, wegen des 1024-Zylinder-Limits etc.), ist die Bootdiskette nutzlos.
- LILO-Bootdiskette mit Kernel: Auf der Diskette wird ein Dateisystem angelegt. Anschließend werden alle LILO-relevanten Daten (`boot.b`, die Kernel-Datei etc.) auf die Diskette kopiert. LILO wird abermals in den MBR der Diskette installiert, findet jetzt aber alle zum Booten relevanten Dateien auf der Diskette. Vorteil: Zuverlässig, kein 1024-Zylinder-Limit. Nachteil: Der Bootprozess dauert um einige Sekunden länger.
- Bootdiskette ohne LILO: Hier wird die Kernel-Datei ohne Dateisystem direkt auf die Sektoren der Diskette geschrieben (beginnend mit dem MBR-Sektor). Beim Booten wird der Kernel ebenso geladen. Diese Variante ist auf Seite 343 beschrieben.

Noch ein Tipp: Obwohl der Konfigurationsaufwand für Variante 2 (LILO mit Kernel) am größten ist, bietet diese Variante die größte Sicherheit, dass Sie Linux in einem unvorhergesehenen Notfall wirklich starten können.

VORSICHT

Im Vergleich zu Festplatten sind Disketten ein sehr fragiles Medium. Sehr viele Probleme mit Bootdisketten haben mit der Verwendung alter Disketten zu tun. Sie sparen sich eine Menge Ärger, wenn Sie neue, möglichst vorformatierte Disketten verwenden!

LILO-Bootdiskette ohne Kernel auf der Diskette

Zum Einrichten benötigen Sie eine `lilo.conf`-Datei nach dem folgenden Muster. Entscheidend ist die erste Zeile mit der `boot`-Option, in der als Ziel für die LILO-Installation der MBR der Diskette angegeben wird. (Da beim Booten nur dieser eine Sektor gelesen wird, ist es überflüssig, auf der Diskette ein bestimmtes Dateisystem anzulegen.)

`/boot/vmlinuz` muss auf einen Kernel zeigen, der alle zum Zugriff auf die Root-Partition erforderlichen Treiber enthält. Pfad- und Device-Angaben müssen Sie an die Gegebenheiten Ihres Rechners anpassen. Unter Umständen benötigen Sie zusätzliche Optionen, die im Konfigurationsabschnitt beschrieben wurden (bei SCSI-Systemen `initrd`, siehe ab Seite 332).

```
# Datei /etc/lilo.conf
boot=/dev/fd0                    # Device des Diskettenlaufwerks
prompt                           # LILO-Eingabe-Prompt anzeigen
timeout=100                      # 10 Sekunden warten
lba32                            # außer bei alten Mainboards
                                 # (vor 1998)
image = /boot/vmlinuz            # Kernel-Datei
        label = linux
        root = /dev/hda8         # Root-Device
        read-only
        # initrd = /boot/initrd  # für SCSI/RAID/LVM/reiserfs etc.
other=/dev/hda1
        label=windows
```

Nachdem Sie die Konfigurationsdatei korrekt eingerichtet haben legen Sie eine formatierte Diskette in das Laufwerk und führen `lilo` aus.

```
root#  lilo
Added linux *
Added windows
```

LILO-Bootdiskette mit eigenem Kernel auf der Diskette

Mit etwas mehr Aufwand ist es möglich, eine LILO-Bootdiskette zu erstellen, die zusätzlich einen eigenen Kernel besitzt. Auf dieser Diskette müssen sich ein Dateisystem (minix oder ext2), die Kernel-Datei und andere für den Bootprozess erforderliche Dateien (`boot.b` und `map`) befinden. Die Vorbereitungsarbeiten sehen folgendermaßen aus:

```
root#  mkfs -t ext2 /dev/fd0 1440
root#  mkdir /floppy
root#  mount -t ext2 /dev/fd0 /floppy
root#  mkdir /floppy/boot
root#  cp /boot/vmlinuz /floppy/boot
root#  cp /boot/boot.b /floppy/boot/
root#  cp /boot/initrd /floppy/boot/
```

`mkfs` legt auf der Diskette ein ext2-Dateisystem an, das über das Verzeichnis `/floppy` angesprochen wird. Das letzte Kommando ist optional und nur bei SCSI/RAID /LVM/reiserfs-Systemen erforderlich.

Der zweite Schritt ist, dass Sie eine eigene LILO-Konfigurationsdatei für die Diskette erstellen. Der wesentliche Unterschied gegenüber der Konfigurationsdatei aus dem obigen

Beispiel besteht darin, dass `map` und `install` jetzt auf Dateien der Diskette verweisen (statt wie sonst üblich auf Festplattendateien).

Die Datei `/floppy/boot/map` wurde in den obigen Kommandos übrigens nicht vergessen. Sie wird durch die Ausführung des `lilo`-Kommandos erzeugt.

Im Beispiel unten sind drei Bootvarianten vorgesehen. In der Defaulteinstellung wird versucht, von der Festplatte zu booten (das geht am schnellsten). Falls es dabei zu einem Absturz kommt (etwa weil die Kernel-Datei nicht mehr da ist, wo sie von LILO erwartet wird), kann ein neuer Versuch gestartet und beim LILO-Prompt die Variante `linuxfromdisk` gewählt werden. Beachten Sie, dass bei dieser Variante `image` auf die Kernel-Datei der Diskette verweist! Als dritte Variante ist – der Vollständigkeit halber – noch das Booten von DOS/Windows möglich.

```
# /etc/lilo.conf-floppy
boot = /dev/fd0                  # Device des Diskettenlaufwerks
prompt                           # LILO-Prompt anzeigen
delay = 100                      # 10 Sekunden warten
install=/floppy/boot/boot.b      # auf der Diskette!
map=/floppy/boot/map             # auf der Diskette!
lba32                            # nicht bei alten Mainboards
                                 #   (vor 1998)
image =  /boot/vmlinuz           # Kernel von der Festplatte
        label = linux
        root = /dev/hda8         # Root-Device
        read-only
image =  /floppy/boot/vmlinuz    # Kernel von der Diskette
        label = linuxfromdisk
        root = /dev/hda8         # Root-Device
        read-only
other = /dev/hda1                # Windows 9x/ME
        label = windows
```

Mit den folgenden Kommandos installieren Sie LILO auf der Diskette. Das `chmod` ist erforderlich, weil sich LILO sonst beschwert.

```
root#  chmod go-w /etc/lilo.conf-floppy
root#  lilo -C /etc/lilo.conf-floppy
root#  ls -lR /floppy/
   drwxrwxr-x   2 root     root           80 Jun 16 20:24 boot

   /floppy/boot:
   total 11
   -rw-rw-r--   1 root     root         3708 Jun 16 19:39 boot.b
   -rw-------   1 root     root         7168 Jun 16 20:24 map
   -rw-rw-r--   1 root     root       403672 Jun 16 19:37 vmlinuz
root#  sync
root#  umount /floppy
```

HINWEIS

Noch mehr Informationen zum Erstellen von ausgeklügelten Bootdisketten, die sogar ein eigenes `root`-Dateisystem besitzen und sich daher für Wartungsaufgaben eignen, finden Sie im HOWTO-Text zum Thema Bootdisketten.

LILO-Installation in den Bootsektor der Festplatte

VORSICHT

Nochmals: Von allen LILO-Varianten ist diese die gefährlichste! Sie kann mit einer Windows-NT/2000/XP-Installation inkompatibel sein. Sie ist auf jedem Fall inkompatibel mit alten Disk-Managern (DOS-Zugriff auf große IDE-Platten bei sehr altem BIOS)!

Sicherungskopie des Bootsektors erstellen

Bei der Ausführung von `lilo` zur Installation in den MBR einer Festplatte wird automatisch getestet, ob sich im Verzeichnis `/boot` bereits eine Sicherheitskopie des Bootsektors befindet. Nur wenn das nicht der Fall ist (also beim ersten Ausführen von `lilo`), kopiert das Kommando den aktuellen Bootsektor in die Datei `/boot/boot.0300` (bei IDE-Platten) oder `/boot/boot.0800` (bei SCSI-Platten). Wenn Sie den Bootsektor manuell sichern möchten, müssen Sie eines der folgenden Kommandos ausführen, *bevor* Sie `lilo` zum ersten Mal ausführen. Das erste Kommando gilt für die erste IDE-Platte, das zweite Kommando gilt für die erste SCSI-Platte:

```
root#  dd if=/dev/hda of=/boot/bootsektor.ide  bs=512 count=1
root#  dd if=/dev/sda of=/boot/bootsektor.scsi bs=512 count=1
```

Wenn Sie den aktuellen Bootsektor auf eine Diskette übertragen möchten, benötigen Sie eine formatierte Diskette (siehe `fdformat` auf Seite 935). Anschließend führen Sie eines der beiden folgenden Kommandos aus (IDE/SCSI):

```
root#  dd if=/dev/hda of=/dev/fd0 bs=512 count=1
root#  dd if=/dev/sda of=/dev/fd0 bs=512 count=1
```

Wenn sich die so erstellte Diskette beim Einschalten des Rechners in Laufwerk A: befindet, bootet der Rechner so wie vom bisher auf der Festplatte befindlichen Bootsektor!

LILO einrichten

Zum Einrichten benötigen Sie eine `lilo.conf`-Datei, die bis auf die `boot`-Zeile so aussieht wie die Diskettenversion. Pfad- und Device-Angaben müssen Sie an die Gegebenheiten Ihres Rechners anpassen. Unter Umständen benötigen Sie zusätzliche Optionen, die im Konfigurationsabschnitt oben beschrieben wurden.

```
# Datei /etc/lilo.conf
boot=/dev/hda                    # Device der ersten IDE-Platte
# boot=/dev/sda                  # Device der ersten SCSI-Platte
prompt
timeout=100
lba32                            # nicht bei alten Mainboards (vor 1998)
image = /boot/vmlinuz            # Kernel-Datei
        label = linux
        root = /dev/hda8         # Root-Device
        read-only
        # initrd = /boot/initrd  # bei SCSI/RAID/LVM/reiserfs-Systemen
other=/dev/hda1
        label=windows
```

Um LILO zu installieren, führen Sie als `root` das Kommando `lilo` aus:

```
root#  lilo
```

LILO von der Festplatte entfernen

Um LILO wieder von der Festplatte zu entfernen, müssen Sie den Bootsektor wiederherstellen. Im einfachsten Fall führen Sie dazu einfach `lilo -u` aus. LILO liest dann aus `/boot` den bei der ersten LILO-Installation gesicherten Bootsektor und überschreibt damit den aktuellen LILO-Bootsektor.

```
root#  lilo -u
```

Wenn Sie eine eigene Sicherheitskopie des Bootsektors angelegt haben, können Sie diesen mit `dd` wieder installieren. Das erste Kommando gilt für die erste IDE-Platte, das zweite Kommando gilt für die erste SCSI-Platte:

```
root#  dd if=/boot/bootsektor.ide of=/dev/hda bs=512 count=1
root#  dd if=/boot/bootsektor.scsi of=/dev/sda bs=512 count=1
```

Falls es bei der Deinstallation von LILO Probleme gibt, können Sie DOS über eine Bootdiskette starten und dort `FDISK /MBR` ausführen. Damit wird ein Bootsektor zum automatischen Start von DOS/Windows eingerichtet (und der LILO-Bootsektor überschrieben). Diese Vorgehensweise ist allerdings bei einigen Windows-9x-Versionen und bei allen NT-Versionen unmöglich!

Das 1024-Zylinder-Limit

Wie bereits erwähnt wurde, ist LILO beim Laden der Kernel-Datei von der Festplatte auf das BIOS angewiesen. Aus historischen Gründen ermöglicht das Standard-BIOS nur eine Adressierung der ersten 1024 Zylinder der Festplatte. Die Zylindergröße ist vom Alter des Mainboard und der Festplatte abhängig; daher kann der Bereich der ersten 1024 Zylinder zwischen 500 MByte und 8 GByte schwanken (siehe Seite 98.)

Neuere Mainboards (ca. seit 1998) sind mit einer BIOS-Erweiterung ausgestattet (den so genannten Extended INT13 Functions). Aktuelle LILO-Versionen (genau genommen seit 21.4) kommen mit dieser Erweiterung zurecht. (Ihre LILO-Version können Sie übrigens mit `lilo -V` ermittteln.)

Wenn Sie also kein zu altes Mainboard haben, gibt es für Sie kein 1024-Zylinder-Limit. Sie müssen lediglich in `lilo.conf` die Option `lba32` angeben.

Aber natürlich gibt es auch mit alten Mainboards viele Möglichkeiten, Linux zu booten:

- Am einfachsten ist es, eine LILO-Bootdiskette mit Kernel zu erstellen (siehe Seite 327).
- Am elegantesten ist es, eine kleine `/boot`-Partition unterhalb der 1024-Zylinder-Grenze anzulegen, in der sich alle für LILO relevanten Dateien befinden. Manche Distributionen schlagen bei der Installation automatisch vor, eine derartige Partition einzurichten. Wenn die ersten 1024 Zylinder der Festplatte allerdings schon mit anderen Partitionen gefüllt sind, ist diese Variante ausgeschlossen. Elegant ist diese Variante deswegen, weil keinerlei Änderungen an `/etc/lilo.conf` notwendig sind, weil direkt von der Festplatte gebootet wird und weil alle Daten von Linux-Partitionen stammen.
- Falls sich innerhalb der ersten 1024 Zylinder eine Windows-3.1/9x/ME-Partition (nicht NTFS!) befindet, können Sie die für LILO relevanten Dateien auch dorthin kopieren. Der Nachteil dieser Variante besteht darin, dass LILO nicht mehr funktioniert, sobald sich der Ort der LILO-Dateien in der Windows-Partition ändert (etwa nachdem die Partition mit einem Programm wie `CHKDSK.EXE` oder `SCANDISK.EXE` defragmentiert wird).

Die folgende Beschreibung erklärt, wie Sie die Kernel-Datei und andere Bootinformationen in einer Windows-Partition unterhalb dieser Grenze unterbringen. Dabei wird angenommen, dass die Partition als `/dev/hda1` angesprochen und vorübergehend als `/winc` in das Linux-Dateisystem eingebunden wird.

```
root#  mkdir /winc
root#  mount -t vfat /dev/hda1 /winc
root#  mkdir /winc/lilo
root#  cp /boot/vmlinuz /winc/lilo
root#  cp /boot/* /winc/lilo
```

In `/etc/lilo.conf` müssen die Pfadangaben zur Kernel-Datei und eventuell zu anderen Dateien entsprechend geändert werden. Neu im Vergleich zu bisherigen `lilo.conf`-Dateien sind die Optionen `install` und `map`, die nicht mehr (wie in der Defaulteinstellung) in das Linux-Verzeichnis `/boot` zeigen, sondern in das Windows-Verzeichnis `C:\LILO`.

Beachten Sie aber, dass die Einstellung für `root` unverändert bleibt! (Die `root`-Option gibt an, wo sich die Linux-Root-Partition befindet. Deren Ort hat sich nicht verändert!)

```
# Datei /etc/lilo.conf
boot=/dev/hda
prompt
timeout=100
install=/winc/lilo/boot.b
map=/winc/lilo/map
image= /winc/lilo/vmlinuz
       root = /dev/hda8        # Linux Root-Partition
       label = linux
       read-only
other=/dev/hda1
        label=windows
```

Jetzt können Sie `lilo` ausführen. Die Windows-Partition muss dabei noch immer gemountet sein. Bei einem Neustart lädt LILO die Kernel-Datei aus der Windows-Partition und startet danach Linux.

VORSICHT

Das Problem bei dieser Art der LILO-Installation besteht darin, dass LILO darauf angewiesen ist, dass sich der Ort der Kernel-Datei nicht ändert. (LILO speichert nicht den Dateinamen, sondern die Sektornummern, auf denen sich die Datei befindet.) Wenn Sie die Windows-Partition defragmentieren oder das `lilo`-Verzeichnis verschieben, kopieren etc., kann sich der Ort der Datei auf der Platte ändern. LILO findet dann die Kernel-Datei nicht mehr und das Booten schlägt fehl. Abhilfe: Linux muss mit einer Bootdiskette gestartet, die Windows-Partition eingebunden und `lilo` abermals ausgeführt werden.

TIPP

Wenn Sie einen neuen Kernel installieren, müssen Sie die Windows-Partition wieder mounten, die neue Kernel-Datei dorthin kopieren und `lilo` neu ausführen. Die Datei `/boot/vmlinuz` spielt im Gegensatz zu einer 'normalen' LILO-Konfiguration keine Rolle – es kommt einzig auf `/winc/lilo/vmlinuz` an!

SCSI/RAID/LVM/reiserfs-Systeme

Vielleicht fragen Sie sich, was der gemeinsame Nenner der Überschrift ist. Nun, alle Begriffe haben mit dem Zugriff auf das Dateisystem zu tun. Und der ist gleich zweimal von Interesse: einmal während der Installation von LILO bei der Ermittlung der Sektorenliste für die Bootdateien und einmal nach dem geglückten Kernel-Start, wenn der Kernel auf das Root-Dateisystem zugreifen möchte.

Je nach Konfiguration kann es sein, dass Sie nur mit einem oder auch mit beiden Problemen konfrontiert sind. Auch die Lösungsansätze (separate `/boot`-Partition, Verwendung einer Initial-RAM-Disk) sind voneinander unabhängig. In den folgenden beiden Abschnitten finden Sie eine genauere Ursachenanalyse.

LILO-Zugriff auf Festplattensektoren

`lilo` muss während der Installation in der Lage sein, dem im Dateisystem gespeicherten Kernel Festplattensektoren zuzuordnen. Die Sektorenliste ist die Basis für den späteren Startprozess.

Wenn der Kernel in einem ext2- oder Windows-9x-Dateisystem gespeichert ist, kann `lilo` die Sektornummern problemlos ermitteln. Wenn Sie andere Dateisysteme verwenden oder wenn zwischen dem Dateisystem und der Festplatte andere Sub-Systeme stehen (LVM oder RAID, siehe Kapitel 6), kann es Probleme geben:

- **Dateisysteme:** Aktuelle LILO-Versionen kommen problemlos mit reiserfs zurecht. Bei älteren LILO-Versionen muss das reiserfs-Dateisystem mit der `mount`-Option `notail` in das Dateisystem eingebunden werden.

  Laut Dokumentation bereitet auch die Kombination zwischen SGI xfs bzw. IBM jfs und LILO kein Problem – ich habe das aber nicht selbst getestet.
- **LVM:** Für das korrekte Zusammenspiel zwischen LVM und LILO gab es erste Patches, als diese Zeilen geschrieben wurden. Es ist daher zu erwarten, dass künftige LVM- und LILO-Versionen miteinander kompatibel sein werden. (Zumindest bis zur LILO-Version 21.7 ist das nicht der Fall.)
- **RAID:** LILO kommt zurzeit ausschließlich mit RAID-1 zurecht (und nur mit den RAID-Tools 0.9, nicht mit der älteren Version 0.4.)
- **SCSI-Festplatten:** Für die Sektorenerkennung spielt es keine Rolle, ob Sie mit IDE- oder SCSI-Festplatten arbeiten, d. h. es gibt keine Probleme.

Wenn es Probleme gibt, ist die Lösung eigentlich einfach: Sie müssen eine kleine `/boot`-Partition mit ext2-Dateisystem ohne LVM und ohne RAID anlegen. Wenn Sie auf Ihrer Festplatte keinen Platz mehr für eine kleine Partition haben (10 MByte reichen), müssen Sie auf eine Bootdiskette mit Kernel zurückgreifen (siehe Seite 327).

LILO-Zugriff auf Festplattensektoren (Initial-RAM-Disk)

Sobald es LILO einmal gelingt, den Kernel zu starten, muss dieser unmittelbar nach dem Start auf das Root-Dateisystem zugreifen können. Der mit den meisten Linux-Distributionen mitgelieferte Standardkernel ist dazu nur in der Lage, wenn Sie IDE-Festplatten und ein ext2-Dateisystem verwenden. Alle Zusatzfunktionen zur Steuerung von SCSI-Controllern, für besondere Dateisysteme, für RAID oder für LVM befinden sich in Modulen, die normalerweise bei Bedarf geladen werden. Noch ist das aber nicht möglich, weil sie ja im Dateisystem gespeichert sind, das noch gar nicht gelesen werden kann.

Es gibt drei Lösungen für das Problem:

- Sie legen die root-Partition auf einer IDE-Festplatte an und verwenden dort ein ext2-Dateisystem ohne LVM oder RAID.
- Sie kompilieren selbst einen Kernel, in den alle Funktionen zum Zugriff auf das Root-Dateisystem bereits integriert sind. Der Nachteil besteht darin, dass das Selbst-

kompilieren oft Probleme bereitet und dass die resultierende Kernel-Datei bisweilen sehr groß wird, was wiederum neue LILO-Probleme verursachen kann.

- Die in den meisten Fällen beste Lösung besteht darin, eine so genannte Initial-RAM-Disk zu verwenden. Der Rest dieses Abschnitts beschränkt sich auf diese Variante.

Ein wenig vereinfacht sieht die Vorgehensweise so aus: Vor der LILO-Installation wird ein kleines Dateisystem erstellt, in dem alle erforderlichen Kernel-Module gespeichert werden. Das gesamte Dateisystem wird in einer einzigen, komprimierten Datei gespeichert. Diese Datei wird in der LILO-Konfigurationsdatei mit der Option `initrd=/boot/initrd` angegeben.

Während des Startprozesses lädt LILO die Datei in den Speicher (RAM), spricht sie wie ein Dateisystem an (eben als RAM-Disk) und stellt sie dem Kernel zur Verfügung. Dieser lädt sich von dort die Module, bevor er dann auf das root-Dateisystem zugreift.

VERWEIS

Exakter und mit mehr Details ist die Verwendung von Initial-RAM-Disks in der folgenden Datei beschrieben (die Teil der Kernel-Codes ist):

`/usr/src/linux/Documentation/initrd.txt`

Der einzig wirklich komplizierte Aspekt bei diesem Verfahren besteht darin, die Datei für die Initial-RAM-Disk herzustellen. Zum Glück helfen dabei eigene Kommandos, die aber je nach Distribution unterschiedlich sind (siehe unten).

Bevor Sie sich mit diesen Kommandos beschäftigen, stellt sich noch die Frage, welche Module Sie eigentlich benötigen. Dazu führen Sie am besten `lsmod` aus und konsultieren `/etc/modules.conf`. Bei der Suche nach den richtigen Modulnamen können Sie auch einen Blick in die Modulverzeichnisse werfen. Die folgenden Verzeichnisangaben sind relativ zu `/lib/modules/`*kernelversion*`/kernel`.

SCSI: `drivers/scsi`.
Dateisysteme: `fs/*/*`.
LVM und RAID: `drivers/md`

HINWEIS

Zur Erzeugung der Initial-RAM-Disk-Datei wird die Datei mit Hilfe eines so genannten Loopback-Device als Dateisystem angesprochen. Damit das klappt, muss der laufende Linux-Kernel diese Funktion unterstützen bzw. das entsprechende Kernel-Modul geladen werden (zur Not manuell: `modprobe loop`).

VERWEIS

Speziell für die LVM-Module gibt es ein eigenes Kommando zur Erzeugung einer Initial-RAM-Disk: `lvmcreate_initrd`. Dieses Kommando berücksichtigt allerdings nur LVM, nicht aber andere, eventuell ebenfalls erforderliche Module. Weitere Informationen gibt die `man`-Seite zu diesem Kommando.

VORSICHT

Bitte denken Sie daran, dass Sie jedes Mal, wenn Sie eine neue RAM-Disk-Datei erzeugen, auch `lilo` neu ausführen müssen. Die neue RAM-Disk-Datei befindet sich nicht auf denselben Sektoren der Festplatten, daher findet die alte LILO-Konfiguration die Datei nicht mehr.

mkinitrd (Mandrake, Red Hat): `mkinitrd` wertet die Datei `/etc/modules.conf` aus und kopiert alle in der Zeile `scsi_hostadapter` angegebenen Module in die RAM-Disk.

```
# in /etc/modules.conf
alias scsi_hostadapter aic7xxx
```

Wenn sich das root-Dateisystem in einer RAID-Partition befindet, werden automatisch auch die hierfür erforderlichen Module in die RAM-Disk kopiert.

Alle weiteren Module – etwa für das Root-Dateisystem oder für LVM – müssen explizit mit der Option `--with=modulname` angegeben werden. (Für jedes Modul ist eine eigene `--with`-Option notwendig.) Die zurzeit aktiven Module können Sie mit `lsmod` bestimmen.

Als weitere Parameter müssen der Name der RAM-Disk-Datei (üblicherweise `/boot/initrd` sowie die Kernel-Version übergeben werden. (Diese kann mit `uname -r` ermittelt werden.) Wenn Sie eine bereits vorhandene RAM-Disk-Datei überschreiben möchten, benötigen Sie außerdem die Option `-f`. Einige weitere Optionen sind im `man`-Test zu `mkinitrd` beschrieben.

Das folgende Kommando erzeugt eine RAM-Disk-Datei für das reiserfs-Dateisystem.

```
root#  mkinitrd -f --with=reiserfs /boot/initrd 2.4.2-2
```

mk_initrd (SuSE): Bei SuSE hat das Kommando nicht nur einen zusätzlichen Unterstrich, auch die Steuerung erfolgt ein wenig anders. Normalerweise müssen an das Kommando keinerlei Parameter oder Optionen übergeben werden. `mk_initrd` wertet die Variable `INITRD_MODULES` in der Datei `/etc/rc.config` aus. Diese Variable enthält nach einer Installation auf einem SCSI-System üblicherweise das benötigte SCSI-Modul. Außerdem erkennt `mk_initrd` automatisch, ob LVM verwendet wird, und fügt bei Bedarf auch dieses Modul hinzu.

`mk_initrd` kümmert sich allerdings weder um RAID-Module noch um Module für zusätzliche Dateisysteme – diese müssen Sie gegebenenfalls selbst in `/etc/rc.config` eintragen. (Sie können die Module auch mit der Option `-m` direkt an `mk_initrd` übergeben.)

Das folgende Beispiel geht davon aus, dass sich Ihr Linux-System in einer reiserfs-Partition auf einer SCSI-Festplatte befindet, die über eine SCSI-Karte von Adaptec angesprochen wird. (Sie können in `INITRD_MODULES` aber natürlich beliebige weitere Module angeben. Achten Sie darauf, dass Sie die Module durch Leerzeichen, nicht durch Kommas trennen!)

```
# in /etc/rc.config
INITRD_MODULES="aic7xxx reiserfs"
```

`mk_initrd` erzeugt nicht nur eine RAM-Disk-Datei, sondern gleich zwei: `/boot/initrd` und `/boot/initrd.suse`. Der Grund besteht darin, dass SuSE per Default zwei gleiche Kernel installiert, `vmlinuz` und `vmlinuz.suse`. Wenn Sie selbst einen eigenen Kernel kompilieren, ersetzen Sie damit `vmlinuz`; `vmlinuz.suse` bleibt aber unverändert. Beim Booten mit LILO können Sie dann entscheiden, ob Sie ihren eigenen Kernel ('linux') oder den von SuSE ('suse') verwenden möchten. (Das ist dann praktisch, wenn Ihnen beim Kompilieren ein Fehler unterläuft.)

```
root#  mk_initrd
using "/dev/hdb11" as root device (mounted on "/")
creating initrd "//boot/initrd" for kernel "//boot/vmlinuz" (2.4.4-4GB)
module aic7xxx is
  "/lib/modules/2.4.4-4GB/kernel/drivers/scsi/aic7xxx/aic7xxx.o"
  -> insmod aic7xxx
module reiserfs is
  "/lib/modules/2.4.4-4GB/kernel/fs/reiserfs/reiserfs.o"
  -> insmod reiserfs

creating initrd "//boot/initrd.suse" for kernel "//boot/vmlinuz.suse"
  (2.4.4-4GB)
module aic7xxx is
  "/lib/modules/2.4.4-4GB/kernel/drivers/scsi/aic7xxx/aic7xxx.o"
  -> insmod aic7xxx
module reiserfs is
  "/lib/modules/2.4.4-4GB/kernel/fs/reiserfs/reiserfs.o"
  -> insmod reiserfs
```

VERWEIS

Weitere Informationen zu `mk_initrd` bekommen Sie mit der Option `-h` oder wenn Sie den Quelltext lesen (Datei `/sbin/mk_initrd`). Es gibt auch in der SuSE-Supportdatenbank `http://sdw.suse.de` einige Artikel zu diesem Thema – suchen Sie nach `initrd`.

LILO durch den Bootmanager von Windows NT/2000/XP starten

Windows NT/2000/XP verwendet einen eigenen Bootmanager, der ähnlich wie LILO funktioniert und normalerweise in den MBR der ersten Festplatte oder der ersten Partition installiert wird. Mit dem Bootmanager können Sie verschiedene Windows-Versionen starten.

HINWEIS

Wenn in diesem Abschnitt einfach von Windows NT die Rede ist, sind die Windows-Versionen NT 4.0, 2000 und XP gemeint. Das Bootsystem dieser Windows-Versionen ist zum Glück identisch.

LILO ist nicht in der Lage, Windows NT selbst zu starten, wenn der von NT vorgegebene Bootsektor durch LILO überschrieben wird. Dieser Abschnitt beschreibt daher den umge-

kehrten Weg: Der Bootmanager von Windows NT bleibt, wo er ist, er wird aber mit einem zusätzlichen Menüeintrag ausgestattet, um LILO zu starten. Beim Rechnerstart können Sie sich also bequem zwischen NT und LILO entscheiden. (Innerhalb von LILO stehen dann unter Umständen abermals mehrere Optionen zur Auswahl, also z. B. verschiedene Linux-Kernel.)

Meiner Meinung nach ist das die bei weitem eleganteste Art, Linux auf einem Windows-NT-System zu booten, weil auch bei Windows-Neuinstallationen und -Updates (Service Packs etc.) keine Probleme zu erwarten sind.

HINWEIS

Der NT-Bootmanager und die dazugehörigen Dateien werden grundsätzlich in die erste Partition der ersten Festplatte installiert. Dieser Abschnitt geht davon aus, dass es sich bei dieser Partition um eine Windows-Partition (3.1/9x/ME) handelt, nicht aber um eine NTFS-Partition.

Die hier beschriebene Vorgehensweise funktioniert selbst dann, wenn die erste Partition eine NTFS-Partition ist. Die Einrichtung von LILO wird dadurch aber etwas umständlicher, weil NTFS-Partitionen unter Linux nicht verändert werden dürfen. (Der NTFS-Treiber sieht zwar einen Read-write-Modus vor, dieser hat aber experimentellen Charakter und sollte nicht verwendet werden.) Sie müssen daher die unten beschriebene Datei `bootsec.lin` auf eine Diskette kopieren, den Rechner unter NT neu starten und dann `bootsec.lin` in die NT-Partition kopieren und `BOOT.INI` verändern.

TIPP

Wenn Sie den NT-Bootmanager versehentlich durch LILO überschrieben haben, können Sie mit `lilo -u` versuchen, den Bootsektor wiederherzustellen. Ist das nicht möglich, müssen Sie den Bootsektor mit dem Installationsprogramm bzw. mit einer Emergency-Diskette von Windows NT wiederherstellen. Aber selbst wenn Sie die erforderlichen Disketten haben, ist das eine mühsame Angelegenheit.

LILO-Konfiguration

Der erste Schritt besteht darin, dass Sie den Bootsektor der Linux-Root-Partition Ihrer Festplatte mit `dd` in eine Datei kopieren. Diese Partition können Sie mit `rdev` ermitteln (im Beispiel unten `/dev/hda8`). Passen Sie bei der Eingabe des `dd`-Kommandos auf! Mit falschen Parametern kann `dd` eine Menge Schaden anrichten!

```
root#  rdev
/dev/hda8 /
root#  dd if=/dev/hda8 bs=512 count=1 of=/boot/bootsec.lin
```

Nun ändern Sie `/etc/lilo.conf` so, dass `lilo` nicht den Bootsektor einer Festplatte oder Diskette verändert, sondern direkt die oben angegebene Datei `/boot/bootsec.lin`. Die restlichen LILO-Einstellungen erfolgen wie bei der Installation von LILO in den MBR der Festplatte (siehe Seite 319 bzw. Seite 329). Wenn Sie das nächste Mal `lilo` ausführen, wird also nur die Datei `bootsec.lin` verändert.

```
# in /etc/lilo.conf
boot=/boot/bootsec.lin
# ... alle anderen Einstellungen wie bisher
```

Der nächste Schritt besteht darin, dem NT-Startprogramm `NTLDR` beizubringen, dass es neben diversen Microsoft-Produkten auch noch Linux gibt. Sämtliche für den Startprozess erforderlichen Dateien befinden sich in der ersten Partition der ersten Festplatte (ganz unabhängig davon, auf welcher Festplatte bzw. Partition NT selbst installiert ist). `bootsec.lin` muss daher ebenfalls in diese Partition kopiert werden.

Wenn es sich bei dieser ersten Partition um eine FAT-Partition handelt (also um ein herkömmliches DOS/Windows-Dateisystem), können Sie diese Partition mit `mount` in das Dateisystem einbinden und `bootsec.lin` einfach in dessen Wurzelverzeichnis kopieren. Wenn es sich dagegen um eine NTFS-Partition handelt, müssen Sie `bootsec.lin` auf eine Diskette kopieren, NT starten und die Datei unter NT von der Diskette in die NTFS-Partition kopieren.

Jetzt müssen Sie nur noch die `NTLDR`-Konfigurationsdatei `BOOT.INI` ändern (erstellen Sie vorher ein Backup!): Diese Datei befindet sich ebenfalls im Wurzelverzeichnis der ersten Partition. Ein Beispiel für diese Datei ist unten abgedruckt. (Im Detail kann die Datei je nach Ihrer Hardware-Konfiguration ein wenig anders aussehen. Lange Zeilen sind hier aus Platzgründen auf zwei Zeilen verteilt, als Trennzeichen wurde \ verwendet. In `BOOT.INI` müssen diese Zeilen allerdings zusammenbleiben!)

```
[boot loader]
timeout=60
default=multi(0)disk(0)rdisk(1)partition(2)\WINNT4
[operating systems]
multi(0)disk(0)rdisk(1)partition(2)\WINNT=\
"Microsoft Windows 2000 Professional" /fastdetect
C:\="Microsoft Windows"
```

An das Ende dieser Datei fügen Sie nun noch eine weitere Zeile an, nämlich:

```
C:\bootsec.lin="LILO"
```

Falls Sie diese Änderung unter Linux durchführen, müssen Sie auf die korrekten Zeilentrennzeichen achten. (Unter DOS/Windows ist ja ein zusätzliches Ctrl-M-Zeichen am Zeilenende üblich. Die Eingabe dieses Zeichens ist mit den meisten Unix-Editoren umständlich (Emacs: Strg+Q, Strg+M). Am einfachsten kopieren Sie eine beliebige Zeile und ändern diese. Beim Kopieren bleibt Ctrl-M am Zeilenende erhalten.)

Wenn Sie Ihren Rechner jetzt neu starten, wird LILO als zusätzliche Zeile neben den bisher schon vorhandenen Betriebssystemen angeführt. Wenn Sie diese Option wählen, startet `NTLDR` den LILO. Dort bestehen dann alle Möglichkeiten von LILO, d. h. je nach Konfiguration können Sie sich jetzt noch zwischen verschiedenen Linux-Kerneln entscheiden.

TIPP

Noch mehr Informationen zum Thema LILO und Windows NT finden Sie im Linux+NT-Loader-Mini-HOWTO. Bei der Veränderung von `BOOT.INI` können Sie sich auch von dem Windows-Programm `BOOTPART.EXE` helfen lassen. Weitere Informationen finden Sie unter:

`http://www.winimage.com/bootpart.htm`

LILO-Fehlermeldungen

Sollte der Start von Linux nicht gelingen, gibt LILO fast immer zumindest einen Hinweis, was die Ursache des Fehlers sein könnte. LILO zeigt während eines vierteiligen internen Startprozesses der Reihe nach die vier Buchstaben 'LILO' an. Gibt es während des Startprozesses Probleme, erscheinen nicht alle Buchstaben – und das erlaubt Rückschlüsse auf die Ursache des Problems. Außerdem zeigt LILO unter Umständen eine Fehlernummer an.

Die folgende Liste gibt zu beiden Varianten einige weitere Informationen. Im Detail sind die Fehlercodes im LILO-Benutzerhandbuch beschrieben (üblicherweise in der Datei `/usr/doc/packages/lilo/user.dvi`).

- **Kein Buchstabe von 'LILO':** LILO wurde vermutlich gar nicht installiert (oder nicht dorthin, wo Sie gedacht hatten.
- **L:** Der erste Teil von LILO konnte geladen werden, nicht aber der zweite. Wahrscheinliche Fehlerursache: LILO hat Probleme, die Festplattengeometrie richtig zu interpretieren. Mögliche Abhilfe: Fügen Sie die Option `linear` in `lilo.conf` ein oder geben Sie die Festplattengeometrie mit `sectors`, `heads` und `cylinders` explizit an.
- **LI:** Auch der zweite Teil von LILO konnte geladen werden, aber beim Ausführen traten Probleme auf. Wahrscheinliche Fehlerursache: abermals Probleme mit der Festplattengeometrie oder `/boot/boot.b` konnte nicht gefunden werden. Wurde die Datei nach der Konfiguration von LILO nochmals verändert? Hat sich die Reihenfolge der Festplatten geändert (z. B. Einbau von `hdb` zwischen `hda` und `hdc`)? Befinden sich Teile der Kernel-Datei außerhalb des 1024-Zylinder-Limits? Abhilfe: Fügen Sie die Option `lba32` in `lilo.conf` ein. Bei einem System, das sowohl SCSI- als auch IDE-Laufwerke enthält, sollten Sie die Optionen `disk` und `bios` ausprobieren. Wenn das nichts hilft, orientieren Sie sich an den obigen Tipps (Fehlermeldung 'L').
- **LIL:** LILO konnte gestartet werden, hat aber Probleme beim Lesen von `/boot/map`. Mögliche Fehlerursache/Behebung: Wie oben, aber für `/boot/map`.
- **LIL?:** Probleme mit `/boot/boot.b`. Abhilfe: Führen Sie `lilo` nochmals aus.
- **LIL-:** Probleme mit `/boot/map`. Abhilfe: Führen Sie `lilo` nochmals aus.
- **LILO:** LILO konnte erfolgreich gestartet werden und hat alle Konfigurationsdateien gefunden.
- **Fehlercode 00:** Problem beim Lesen der Sektoren der Kernel-Datei. Mögliche Ursachen: Die Kernel-Datei wurde nach der Installation von LILO verändert (z. B. Neukompilierung), ohne `lilo` neu auszuführen. Oder Sie haben die Option `linear` ver-

wendet und dabei das 1024-Zylinder-Limit überschritten. (Wenn diese Option verwendet wird, erkennt LILO die Überschreitung dieses Limits bei der Installation unter Umständen nicht.) Abhilfe: Kopieren Sie die Kernel-Datei in eine Partition, die vollständig unterhalb der 1024-Zylinder-Grenze liegt, und installieren Sie LILO neu.

- **Fehlercode 01:** Auch wenn das LILO-Benutzerhandbuch behauptet, dass dieser Fehler gar nicht auftreten sollte, ist er einer der häufigsten. Normalerweise erscheint eine endlose Reihe von 01-Codes, bis der Rechner neu gestartet wird. LILO teilt damit mit, dass ein oder auch viele unzulässige Kommandos ausgeführt wurden.

  Eine Ursache kann darin bestehen, dass LILO die Festplatte nicht findet, auf der sich die Kernel-Datei befindet (insbesondere, wenn sich der Kernel nicht auf der ersten Festplatte befindet). In diesem Fall kann die explizite Angabe der Festplattennummer durch die `bios`-Option weiterhelfen (z. B. `bios=0x81` für die zweite Platte).

  Einmal ließ sich das Problem lösen, indem die Option `message=...` auskommentiert wurde, die eine Linux-Distribution per Default vorgesehen hatte. (Auf den Begrüßungstext kann man gern verzichten.) Aber auch alle anderen oben schon erwähnten Tipps (Option `lilo`, explizite Angabe der Festplattengeometrie etc.) können vielleicht helfen.
- **Fehlercode 02:** 'Address mark not found': Eine mögliche Ursache ist eine defekte Diskette. Installieren Sie LILO auf eine neue Diskette.
- **Fehlercode 04:** 'Sector not found': Abermals ist ein Geometrieproblem die wahrscheinlichste Ursache. Abhilfe wie oben. Falls Sie `compact` verwenden, kommentieren Sie diese Option aus.

TIPP

LILO kennt noch eine Reihe weiterer Fehlercodes. Hier wurden nur die häufigsten dokumentiert. Eine vollständige Liste finden Sie in der schon erwähnten LILO-Dokumentation (Datei `user.dvi` bzw. `user.ps`).

Default-Betriebssystem für einen Reboot einstellen

Wenn Sie Ihren Rechner herunterfahren (z. B. durch Strg+Alt+Entf), wird beim nächsten Start wieder LILO ausgeführt. Manchmal wissen Sie beim Herunterfahren, wie Sie den Rechner neu starten möchten – z. B. abermals mit Linux oder mit Windows. In diesem Fall können Sie vor dem Shutdown `lilo -R name` ausführen. Damit erreichen Sie, dass das mit *name* bezeichnete Betriebssystem beim nächsten LILO-Start automatisch (ohne Benutzereingabe) gestartet wird. Der angegebene Name muss einer `label`-Zeichenkette in `/etc/lilo.conf` entsprechen.

Die zwei folgenden Kommandos bewirken, dass der Rechner unter Windows neu gestartet wird (sofern es in `lilo.conf` eine Bootvariante mit dem Namen `windows` gibt):

```
root#  lilo -R windows
root#  shutdown -R now
```

Kurz zu den Interna: Das Defaultsystem wird in der Mapping-Datei gespeichert (üblicherweise also in `/boot/map`). Unmittelbar nachdem LILO die Zeichenkette dort beim

nächsten Start gelesen hat, wird die Zeichenkette gelöscht. Damit wird sichergestellt, dass die Einstellung nur einmal gilt. (Das ist eine Sicherheitsmaßnahme. Wenn das durch *name* angegebene Betriebssystem sich aus irgendeinem Grund nicht starten lässt, so funktioniert LILO nach dem erfolglosen Startversuch wieder normal, reagiert also wieder auf Benutzereingaben.)

`lilo -R` kann dazu eingesetzt werden, um den Komfort bei einem Rechnerneustart zu vergrößern. Insbesondere der KDE-Display-Manager `kdm` benutzt dieses Merkmal (siehe auch Seite 533).

Menüs und Grafikmodus

Bis jetzt ist es in diesem Abschnitt nur darum gegangen, LILO zum Laufen zu bringen. Wenn das geklappt hat und Sie noch Lust auf weitere Experimente haben, können Sie nun versuchen, LILO mit einer Begrüßungsgrafik zu verschönern und die Bedienung mit einem Menü zu erleichern.

LILO-Menü im Textmodus

Ein Menü zur Auswahl des Betriebssystems, das mit den Cursortasten bedient werden kann, ist einfach zu bewerkstelligen. Sie müssen lediglich statt `/boot/boot.b` die Datei `/boot/boot-menu.b` verwenden. Als Menütexte werden die `label`-Texte der einzelnen Betriebssysteme verwendet.

Die Überschrift des Menüs sowie die Farbgestaltung können mit den `lilo.conf`-Optionen `menu-title` und `menu-scheme` eingestellt werden (Details siehe `man lilo.conf`). Beim folgenden Beispiel erscheint das LILO-Menü in einer weißen Box. Der Text wird schwarz angezeigt, das ausgewählte Betriebssystem rot, die Überschrift und der Rahmen um das Menü blau.

```
# in /etc/lilo.conf
install=/boot/boot-menu.b
menu-title="Waehlen Sie ein Betriebssystem aus!"
menu-scheme=kw:Wr:bw:bw
... wie bisher
```

LILO-Menü im Grafikmodus

Hier gibt es zurzeit zwei Ansätze (Mandrake und SuSE). es ist noch nicht abzusehen, ob sich einer davon als Standard durchsetzen wird. Leider sind beide Varianten gleichermaßen schlecht dokumentiert. Zudem ist die Konfiguration sehr aufwändig, sodass sich der Aufwand wohl selten lohnt.

Der gemeinsame Nenner beider Varianten besteht darin, dass eine speziell präparierte Datei mit der `lilo.conf`-Option `message` angegeben wird.

Mandrake 8 verwendet LILO 21.7 mit `boot-graphic.b`. Die Datei mit dem Hintergrundbild muss mit dem Perl-Script `bmp2mdk` aus einer BMP-Datei erstellt werden. Dieses Script wird mit dem LILO-Paket mitgeliefert. Sie finden es samt der Datei `README.graphic` im Verzeichnis `/usr/share/doc/lilo-0.21.7`.

```
# in /etc/lilo.conf
install=/boot/boot-graphic.b
message=/boot/bmp2mdk-file
... wie bisher
```

Weitere Informationen zur Mandrake-LILO-Konfiguration finden Sie hier:

`http://liquid2k.com/matthias/linux/lilo.html`

SuSE 7.2 verwendet noch LILO 21.6 und setzt `boot-menu.b` ein. Eine für LILO geeignete Grafik wird mit `mklilomsg` erzeugt und ebenfalls via `message=file` übergeben. `mklilomsg` erwartet die Ausgangsdatei im PCX-Format. Die PCX-Datei muss bereits alle Menütexte enthalten! Die Menütexte für die verschiedenen Betriebssysteme können bei dieser Variante also nicht mehr mit der LILO-Option `label` verändert werden. (Die genaue Position des Menüs übergeben Sie mit unzähligen Parametern an `mklilomsg`.) Ein weiterer Unterschied zur Mandrake-Lösung besteht darin, dass der VGA-Mode hier über die LILO-Option `vga` eingestellt wird.

```
# in /etc/lilo.conf
vga     = 771
install=/boot/boot-menu.b
message = /boot/mklilomsg-file
... wie bisher
```

Weitere Informationen zur SuSE-LILO-Konfiguration finden Sie hier:

`http://sdb.suse.de/de/sdb/html/jkoeke_bootgrafik.html`
`http://www.13thfloor.at/Software/lilo-splash/Example/`

LILO-Konfigurationshilfen

Anstatt `/etc/linux.conf` selbst zu erstellen, können Sie dazu diverse Hilfsprogramme verwenden. In der Vergangenheit hatten diese Tools aber immer Probleme mit den in diesem Kapitel beschriebenen Sonderfällen (SCSI-Systeme, 1024-Zylinder-Limit, Windows NT/2000), weswegen ich meine LILO-Konfigurationsdateien nach wie vor am liebsten selbst editiere.

Distributionsspezifische Werkzeuge: Die folgende Liste zählt einige Werkzeuge zur Erstellung von Bootdisketten und zur LILO-Installation auf.

Mandrake: `drakfloppy`, `drakboot`, `mkbootdisk` (siehe Seite 1198)
Red Hat: `mkbootdisk` (siehe Seite 1216)
SuSE: YaST (siehe Seite 1236)

Linuxconf: Bei Linuxconf ist die Konfiguration von `lilo.conf` über mehrere Dialoge verteilt (Startpunkt: BOOT MODE|LILO). Als Installationsort kann eine beliebige Partition, der MBR der Festplatte (geben Sie `/dev/hda` bzw. bei SCSI-Systemen `/dev/sda` an) oder eine Diskette ausgewählt werden. Das Ergebnis ist in letzterem Fall eine LILO-Bootdiskette ohne Kernel auf der Diskette.

KLILO (KDE): Bei manchen Distributionen wird `klilo` als weitere Alternative mitgeliefert. Das Programm wurde aber in letzter Zeit nicht mehr gewartet.

8.2 LILO-Alternativen (SYSLINUX, Loadlin)

Es muss nicht immer LILO sein! Dieser Abschnitt zeigt, wie Sie eine Bootdiskette ohne irgendeinen Boot-Loader erstellen können. Außerdem werden drei Alternativen zu LILO kurz vorgestellt: SYSLINUX, Loadlin und (im nächsten Abschnitt) GRUB.

Bootdiskette ohne Boot-Loader

Um Linux mit einer Bootdiskette zu starten, benötigen Sie weder LILO noch eine der in diesem Abschnitt beschriebenen Alternativen. Dazu wird eine etwas veränderte Kernel-Datei direkt in den Sektoren der Diskette gespeichert (beginnend mit dem Bootsektor).

Die Vorteile der hier vorgestellten Variante: geringer Konfigurationsaufwand, keine Probleme mit dem 1024-Zylinder-Limit. Nachteil: inflexibel, d. h. kein LILO-Prompt, keine RAM-Disk (d. h. nur dann für SCSI-, RAID-, LVM-, reiserfs-Systeme geeignet, wenn Sie sich selbst einen maßgeschneiderten Kernel kompilieren).

```
root#  fdformat /dev/fd0            # Diskette formatieren (falls unformatiert)
root#  cp /boot/vmlinuz /dev/fd0    # Kernel dorthin kopieren
root#  rdev                         # Bootpartition ermitteln
/dev/sdc2 /
root#  rdev /dev/fd0 /dev/sdc2      # diese Partition im Kernel einstellen
root#  rdev -R /dev/fd0 1           # Bootpartition zuerst read only anmelden
root#  rdev -v /dev/fd0 -2          # optional: erweiterter VGA-Modus
```

In den obigen Zeilen gehe ich davon aus, dass die Kernel-Datei den Namen `/boot/vmlinuz` hat. Statt `cp` können Sie übrigens auch das in vielen anderen Linux-Büchern angegebene Kommando `dd` verwenden, um die Kernel-Datei auf die Diskette zu übertragen (Seite 931). Beide Kommandos funktionieren für diese Anwendung gleich: Sie übertragen die Kernel-Datei direkt in die ersten Sektoren der Diskette. Auf der Diskette befindet sich damit kein Dateisystem, sondern nur eine Serie von Datenblöcken. (Falls sich vor dem `cp`- oder `dd`-Kommando ein Dateisystem auf der Diskette befand, wird es überschrieben.) Auf dieser Diskette kann keine zweite Datei mehr gespeichert werden, ohne die erste zu zerstören.

Entscheidend bei der Erstellung einer Bootdiskette ist, dass mit `rdev` die korrekte Root-Partition angegeben wird (sonst kann zwar der Kernel geladen werden, der laufende Kernel findet aber die Partition mit dem Linux-Dateisystem nicht). Die korrekte Bezeichnung der Root-Partition (im obigen Beispiel also `/dev/sdc2`) können Sie am leichtesten feststellen, wenn Sie `rdev` ohne Parameter aufrufen. Alternativ gibt `df` an, welche Dateisysteme sich auf welchen Partitionen befinden.

SYSLINUX

Mit SYSLINUX können Sie eine DOS-formatierte Diskette zum Linux-Start verwenden. Der wesentliche Vorteil von SYSLINUX besteht darin, dass sowohl die Konfigurationsdateien als auch alle anderen Dateien auf der Diskette unter Windows verändert werden können. Mit anderen Worten: Wenn Linux aus irgendeinem Grund nicht mehr gestartet werden kann, können Sie Ihren (oder einen anderen) Rechner unter DOS oder Windows starten und dann die Konfigurationsdatei auf der SYSLINUX-Diskette ändern etc.

Praktisch ist auch, dass beim SYSLINUX-Start wie bei LILO zusätzliche Kernel-Optionen angegeben werden können und dass es sehr einfach ist, den Funktionstasten unterschiedliche Hilfetexte zuzuordnen. Deswegen verwenden relativ viele Distributionen SYSLINUX für ihre Installationsdisketten.

Die folgenden Zeilen zeigen, wie Sie eine SYSLINUX-Bootdiskette unter Linux erstellen können. Dabei wird vorausgesetzt, dass die Diskette DOS-formatiert und leer ist. `syslinux` kopiert den SYSLINUX-Bootsektor auf die Diskette. (Es gibt übrigens auch ein `SYSLINUX.EXE`-Programm, das dieselbe Aufgabe unter DOS/Windows erledigt.) Der Default-Linux-Kernel auf der Diskette muss den Namen `linux` haben.

```
root#  syslinux /dev/fd0
root#  mcopy /boot/vmlinuz a:linux
root#  mcopy /boot/initrd a:
```

Jetzt müssen Sie noch die Konfigurationsdatei `syslinux.cfg` erstellen. Die Datei muss das folgende Aussehen haben. `prompt` und `timeout` bewirken, dass für zehn Sekunden ein Eingabeprompt angezeigt wird, bevor das Default-Betriebssystem gestartet wird. Linux-Betriebssysteme werden durch die drei Parameter `label` (Name), `kernel` (Dateiname der Kernel-Datei) und `append` (Kernel-Optionen) angegeben. Im Beispiel wird ein Kernel mit Initial-RAM-Disk geladen.

```
prompt 1
timeout 100
default linux
label linux
  kernel linux
  append initrd=initrd root=/dev/hdb10
```

Kopieren Sie auch diese Datei auf die Diskette:

```
root#  mcopy syslinux.cfg a:
```

VERWEIS

Weitere Informationen zu SYSLINUX und zu seinen Varianten ISOLINUX (für Boot-CDs) und PXELINUX (zum Booten über ein Netzwerk) finden Sie hier:

`http://syslinux.zytor.com/`

LOADLIN (Linux von DOS aus starten)

Mit dem Programm `LOADLIN.EXE` können Sie Linux von DOS aus starten. Durch eine entsprechende Konfiguration von `AUTOEXEC.BAT` kann DOS so eingerichtet werden, dass Sie nach dem Booten zwischen Linux und DOS wählen können. Der größte Vorteil von `LOADLIN` besteht darin, dass Sie bei der Installation den Bootsektor nicht anrühren müssen.

Der Nachteil besteht darin, dass DOS als Betriebssystem zur Verfügung stehen muss. Diese Voraussetzung ist nur erfüllt, wenn Sie mit DOS, Windows 3.1 oder Windows 9x/ME arbeiten. Bei Rechnern, auf denen nur Windows NT/2000/XP installiert ist, kann `LOADLIN` überhaupt nicht verwendet werden.

Vorbereitungsarbeiten

Die Verwendung von `LOADLIN` setzt voraus, dass Sie den Device-Namen Ihrer Root-Partition kennen. Wenn Sie nicht sicher sind, führen Sie `df` aus und schreiben Sie den Device-Namen des Dateisystems auf, das an der Stelle `/` gemountet ist (das wäre im folgenden Beispiel `/dev/sda15`).

```
root#  df
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/sda15            303251  261723    25867     91%   /
/dev/scd0             584560  584560        0    100%   /cdrom
/dev/sda12            303251  114286   173304     40%   /home1
/dev/sda11            303251  189963    97627     66%   /usr/local
```

Des Weiteren benötigen Sie eine Kopie der Kernel-Datei, die unter DOS gelesen werden kann. Sie müssen die Datei also in eine DOS- oder Windows-Partition kopieren. Am einfachsten binden Sie dazu die Partition C: in das Linux-Dateisystem ein und kopieren die Kernel-Datei (zumeist `/boot/vmlinuz`) dorthin. Ein sinnvoller Ort ist das Verzeichnis `C:\LOADLIN`, das Sie gegebenenfalls unter Linux anlegen können. Bei `mount` müssen Sie statt `/dev/sda1` den Namen der ersten DOS-Partition angeben. Wenn Sie den Namen nicht wissen, führen Sie `fdisk -l` aus.

```
root#  mkdir /winc
root#  mount -t msdos /dev/sda1 /winc
root#  mkdir /winc/loadlin
root#  cp /boot/vmlinuz /winc/loadlin
root#  cp /cdrom/dosutils/loadlin.exe /winc/loadlin
root#  umount /winc
```

Loadlin manuell aufrufen

Damit sind die Vorbereitungsarbeiten unter Linux abgeschlossen. Verlassen Sie Linux mit (Strg)+(Alt)+(Entf) und booten Sie DOS. (Es reicht nicht, unter Windows ein DOS-Fenster zu öffnen! Bei Windows 95 können Sie sofort unter DOS booten, indem Sie beim Booten (F8) drücken und die entsprechende Option wählen.) Unter DOS können Sie `loadlin` jetzt testen:

```
> C:
> CD \LOADLIN
> SMARTDRV /C
> LOADLIN vmlinuz root=/dev/sda15 ro
```

VORSICHT

Beachten Sie, dass Sie nach dem Start von Linux nicht mehr zurück zu DOS kommen. Speichern Sie vorher alle eventuell noch geöffneten Dateien, beenden Sie Windows etc. Falls Sie Windows 3.1 verwenden, bewirkt das Kommando `SMARTDRV /C`, dass alle eventuell noch offenen Schreiboperationen auf der Festplatte durchgeführt werden.

VERWEIS

Der manuelle Aufruf von Loadlin ist eher unbequem. Eine erste Verbesserung könnte darin bestehen, dass Sie eine Batch-Datei schreiben, die den `loadlin`-Start mit allen Parametern erledigt. Noch eleganter ist es freilich, wenn gleich beim Start des Rechners ein Menü erscheint, in dem Sie zwischen DOS/Windows und Linux auswählen können. Eine genaue Beschreibung, wie Sie ein derartiges Menü einrichten können, finden Sie im Loadlin+Win95/98/ME-mini-HOWTO.

8.3 GRUB

GRUB steht für *GRand Unified Bootloader*. GRUB bietet fast alle Funktionen, die auch LILO bietet: Menü-Steuerung, Hintergrundbild etc. Außerdem hat das Programm keine Probleme mit dem 1024-Zylinder-Limit (d. h.eigentlich bot es schon lange vor LILO eine Lösung zu diesem Problem).

Darüber hinaus bietet GRUB eine Reihe von Vorteilen gegenüber LILO:

- GRUB ist eine Art Minibetriebssystem, das interaktiv bedient werden kann und das mehrere Dateisysteme (darunter ext2, reiserfs und vfat) direkt unterstützt. Aus diesem Grund ist es nicht erforderlich, nach jeder Änderung des Kernels oder der Initial-RAM-Disk-Datei GRUB neu zu installieren – GRUB findet die Bootdateien selbstständig, solange sich ihr Dateiname nicht ändert.

  Selbst wenn die Kernel-Datei (etwa nach der Installation einer neuen Version) einen neuen Namen bekommen hat, stellt das kein unüberwindbares Hindernis dar: Der Anwender muss den Dateinamen eben nach dem GRUB-Start selbst angeben. Nach der Angabe der Root-Partition hilft GRUB dabei durch die automatische Ergänzung von Dateinamen mit (Tab). (Bei LILO wären Sie in dieser Situation bereits verloren.)

Noch deutlicher formuliert: Wenn Sie sich eine GRUB-Diskette erstellen, auf der sich alle GRUB-Dateien befinden (eine Anleitung folgt weiter unten), können Sie damit mit etwas Probieren jedes Linux-System booten, bei dem sich die Kernel- und Initial-RAM-Disk-Dateien in ext2- oder reiserfs-Partitionen befinden.

- GRUB kann neben Linux und Windows auch einige andere freie Unix-Systeme starten (z. B. BSD-Varianten und GNU Hurd). GRUB bietet sich also besonders dann an, wenn Sie auf einem Rechner mehrere derartige Betriebssysteme parallel installiert haben.
- GRUB enthält Netzwerkfunktionen, die es ermöglichen, die zum Booten erforderlichen Daten über ein Netzwerk zu laden.

Leider ist GRUB aber auch mit Nachteilen verbunden:

- Die Dokumentation ist bei weitem nicht so klar und umfassend wie bei LILO.
- Es sind noch keine komfortablen Konfigurationswerkzeuge verfügbar.
- Die GRUB-Konfiguration ist schwieriger als die von LILO.
- Es gibt keine einfache Möglichkeit, das Tastaturlayout zu verändern. Deswegen gilt bei der Verwendung von GRUB im interaktiven Modus üblicherweise das US-Tastaturlayout. (GRUB sieht ein Kommando vor, mit dem *einzelnen* Tasten andere Zeichen zugeordnet werden können. Es ist aber schwierig, damit das gesamte Tastaturlayout zu ändern.)
- Wie LILO hat auch GRUB Probleme mit Windows NT/2000/XP. Ob ein Start dieser Betriebssysteme gelingt, hängt von deren Installation ab. Auf einem meiner Testrechner, auf dem neben verschiedenen Linux-Distributionen auch Windows NT 4 und Windows 2000 installiert war, gelang es GRUB nicht, eines der beiden Windows-Betriebssysteme direkt zu starten oder auch nur den Windows-NT-Bootloader auszuführen.
- GRUB sieht keine De-Installation vor. Bei der Installation werden die betreffenden Sektoren der Festplatte überschrieben, ohne eine Sicherheitskopie zu erstellen.

VORSICHT

Die zwei letzten Punkte sollten eigentlich schon klar machen, dass die in diesem Buch schon vielfach wiederholte LILO-Warnung auch für GRUB gilt: Lassen Sie nie ein Installationsprogramm GRUB in den Bootsektor (MBR) der Festplatte installieren! Wenn Sie Pech haben, können Sie danach Windows nicht mehr starten.

Ideal ist eine Installation auf eine Diskette. Die Installationsprogramme mancher Distributionen (z. B. Red Hat 7.2) bieten diese Option aber leider nicht an. Stattdessen gibt es die Option, GRUB in den Bootsektor der Linux-Root-Partition zu installieren (z. B. in den Bootsektor von `/dev/hda7`). Das ist zwar ungefährlich, hilft aber nur dann weiter, wenn Sie später mit einem anderen Programm diese Partition als Bootpartition auswählen können oder wenn Sie diese Partition – z. B. mit `fdisk` – als Default-Bootpartition deklarieren. (Normalerweise gilt `/dev/hda1` als Default-Bootpartition.)

GRUB wurde bis vor kurzem nur von wenigen Distributionen (vor allem Mandrake) unterstützt und hat sich trotz seiner zum Teil überlegenen Funktionen unter Linux nicht so

recht durchsetzen können. Das könnte sich nun aber ändern, weil Caldera mit Version 2.4 und Red Hat mit Version 7.2 von LILO auf GRUB als Default-Bootloader umgestiegen sind. Es ist möglich, dass andere Distributionen diesem Vorbild folgen. (LILO wird meistens weiterhin als zweite Bootvariante unterstützt. Wenn Sie die Wahl haben, empfehle ich Ihnen LILO, weil das Programm ausgereifter ist und weil Sie im Fall von Problemen eher Hilfe bekommen. Aber derartige Empfehlungen sind natürlich immer subjektiv.)

VERWEIS

Dieser Abschnitt bezieht sich auf Version 0.90. Aus Platzgründen werden bei weitem nicht alle Möglichkeiten von GRUB beschrieben, sondern nur die wichtigsten Kommandos und Installationsvarianten vorgestellt.

Die offizielle, aber leider ziemlich unübersichtliche GRUB-Dokumentation können Sie mit `info grub` lesen. Weiters gibt es ein Multiboot-with-GRUB-Mini-HOWTO sowie eine FAQ auf der GRUB-Homepage:

`http://www.gnu.org/software/grub/`

Die besten GRUB-Informationen, die ich im Internet entdecken konnte, befinden sich hier:

`http://www.caldera.com/SxS/grub.htm`
`http://www.linuxgazette.com/issue64/kohli.html`

VORSICHT

Mit den `make`-Dateien des Kernel-Codes wird bei einer Neuinstallation des Kernels unter Umständen automatisch auch LILO neu installiert. Deswegen kann es passieren, dass bei einem Kernel-Update GRUB durch LILO überschrieben wird. Entfernen Sie in solch einem Fall einfach das LILO-Paket von Ihrem Rechner!

Anwendung

Dieser Abschnitt geht davon aus, dass GRUB bereits erfolgreich installiert wurde (z. B. im Rahmen einer Linux-Installation) und nach dem Neustarten des Rechners am Bildschirm erscheint. Normalerweise ist dann ein Menü zu sehen, aus dem Sie mit den Cursor-Tasten ein Betriebssystem auswählen und mit ↵ starten können. All das kennen Sie ja bereits von LILO.

Dass es sehr wohl große Unterschiede gibt, erkennen Sie, wenn Sie versuchen, einen Eintrag des GRUB-Menüs zu verändern (mit E wie *edit*) oder das Menü ganz verlassen (mit C wie *command line*). Sie finden sich nun in einer Art Shell wieder, in der Sie über die Tastatur Kommandos eingeben können. Eine Kommandoübersicht gibt `help`. Nähere Informationen zu einem Kommando erhalten Sie mit `help kommando`. Einige Kommandos werden auch im Verlauf dieses Abschnitts vorgestellt. Mit Esc gelangen Sie jederzeit zurück in das Bootmenü.

HINWEIS

GRUB kann durch ein Passwort abgesichert sein. In diesem Fall können Sie die interaktiven Funktionen von GRUB erst verwenden, nachdem Sie (P) gedrückt und dann das Passwort angegeben haben.

Unter GRUB gilt normalerweise das US-Tastaturlayout. Falls Sie mit einer deutschen Tastatur arbeiten, finden Sie auf Seite 97 eine Tabelle zur Eingabe wichtiger Sonderzeichen. Beachten Sie auch, dass (Y) und (Z) vertauscht sind.

Linux-Kernel-Bootoptionen übergeben: Wie bei LILO können Sie auch bei GRUB vor dem Start zusätzliche Kernel-Parameter angeben. Dazu wählen Sie den betreffenden Menüeintrag aus und begeben sich mit (E) in den Edit-Modus. Es werden nun einige Zeilen angezeigt, die so ähnlich wie das folgende Muster aussehen:

```
root (hd1,12)
kernel /boot/vmlinuz-2.4.6-3.1 ro root=/dev/hdb13
initrd /boot/initrd-2.4.6-3.1.img
```

Wählen Sie mit den Cursor-Tasten die `kernel`-Zeile aus und drücken Sie abermals (E), um nun diese Zeile zu verändern, und fügen Sie an das Ende dieser Zeile die zusätzlichen Kernel-Parameter hinzu. Mit (↵) bestätigen Sie die Änderung. (Esc) führt zurück zum Bootmenü, wo Sie Linux dann starten. (Die Änderung an den Kernel-Parametern gilt nur für dieses eine Mal, sie wird also nicht bleibend gespeichert.)

Interaktiv Kommandos ausführen: Wenn Sie den interaktiven Modus starten, können Sie die oben aufgezählten Kommandos (`root`, `kernel` etc.) auch manuell ausführen. Wenn Sie anschließend noch `boot` ausführen, wird das so ausgewählte Linux-Betriebssystem tatsächlich gestartet. (Für andere Betriebssysteme variiert die Kommandoabfolge ein wenig.) Dabei ist interessant, dass GRUB für das durch `root` ausgewählte Dateisystem mit (Tab) Dateinamen vervollständigt. Mit `cat` können Sie einzelne Dateien sogar anzeigen. Daneben gibt es noch viele andere Möglichkeiten, auf die hier aber nicht eingegangen wird.

Menü bleibend verändern: GRUB liest das Bootmenü je nach Installation (d. h. je nach Distribution!) aus der Datei `/boot/grub/grub.conf`, `/boot/grub/menu.lst` oder einer Datei mit einem ähnlichen Namen.

Es ist leider nicht möglich, den Ort der Menüdatei nach erfolgter GRUB-Installation festzustellen (das heißt, es gibt keine Konfigurationsdatei, die den Ort der Menüdatei festschreibt). Die Datei muss auf jeden Fall Kommandos wie `title`, `root`, `kernel`, `chainloader` etc. enthalten. Wenn Sie also das Menü verändern möchten, müssen Sie Linux starten, die GRUB-Menüdatei suchen und verändern. GRUB berücksichtigt Ihre Änderungen automatisch ab dem nächsten Start. (Weitere Informationen zum Aufbau der Menüdatei folgen weiter unten.)

GRUB versus LILO (Interna)

Wenn Sie bisher mit LILO gearbeitet haben, müssen Sie vollkommen umdenken. Bei LILO gibt es zwei Programme: Mit dem Kommando `lilo`, das Sie während der Installation unter Linux ausführen, installieren Sie anhand einer Konfigurationsdatei den Bootloader auf eine Diskette oder in den Bootsektor einer Festplatte. Beim Rechnerstart wird der Bootloader (der ebenfalls LILO heißt, aber ganz andere Funktionen als das Kommando `lilo` bietet) ausgeführt.

Bei GRUB gibt es dagegen nur ein Programm! Wenn Sie unter Linux arbeiten, können Sie das Kommando `grub` ausführen. Sie gelangen damit in eine interaktive Shell, die Sie unter anderem zur Installation von `grub` in den Bootsektor einer Festplatte oder Partition verwenden können. Beim Rechnerstart wird exakt dasselbe Programm `grub` ausgeführt. Auch jetzt können Sie diverse Administrationskommandos ausführen – aber im Regelfall werden Sie einfach nur ein Betriebssystem starten wollen.

Um die Unterschiede zusammenzufassen:

- `lilo` ist ein Kommando zur Installation von LILO auf eine Diskette oder Festplatte.
- LILO ist ein winziges Programm zum Start von Linux oder Windows. Das Programm ermöglicht nur minimale interaktive Eingriffe (Auswahl des Betriebssystems, Angabe zusätzlicher Kernel-Optionen).
- GRUB ist ein komplexes Programm (eigentlich ein Mini-Betriebssystem), mit dem Sie einerseits menügesteuert oder interaktiv unterschiedliche Betriebssysteme starten, andererseits aber auch die gesamte GRUB-Installation durchführen können.

Bezeichnung von Festplatten und Partitionen

GRUB kennt eine eigene Nomenklatur zur Bezeichnung von Festplatten und den darauf enthaltenen Partitionen. Die Grundregel lautet, dass die Nummerierung immer mit null beginnt.

GRUB-Partitionsnamen	
`(hd0)`	die erste Festplatte (entspricht `/dev/hda`)
`(hd0,0)`	die erste Partition der ersten Festplatte (`/dev/hda1`)
`(hd2,7)`	die achte Partition der dritten Festplatte
`(fd0)`	das Diskettenlaufwerk

HINWEIS

Beachten Sie, dass GRUB CD-Laufwerke bei der Nummerierung nicht berücksichtigt!

SCSI-Festplatten werden wie IDE-Festplatten bezeichnet. Wenn sich in einem Rechner sowohl IDE- als auch SCSI-Festplatten befinden, erhalten zuerst die IDE- und dann die SCSI-Festplatten fortlaufende Nummern. (Bei zwei IDE-Festplatten bekommt die erste SCSI-Festplatte daher den Namen `(hd2)`.)

GRUB-Minimalinstallation auf einer Diskette

Um GRUB auf eine Diskette zu installieren, reichen die drei folgenden Kommandos aus. Die Pfadangabe kann je nach Distribution variieren. Verwenden Sie aber nicht die Dateien aus `/boot/grub`, sondern die aus dem GRUB-Installationsverzeichnis! Die `seek`-Option beim zweiten `dd`-Kommando bewirkt, dass der erste Sektor der Diskette mit `stage1` durch das zweite Kommando nicht überschrieben wird.

```
root#  cd /usr/share/grub/i386-pc
root#  dd if=stage1 of=/dev/fd0 bs=512
root#  dd if=stage2 of=/dev/fd0 bs=512 seek=1
```

Wenn Sie Ihren Rechner nun mit dieser Diskette starten, wird GRUB ausgeführt. Mangels Menüs aktiviert GRUB den interaktiven Modus. Sie können nun GRUB-Kommandos ausführen. Um Ihr Linux-System zu starten, geben Sie die folgenden Kommandos ein. (Das Beispiel geht davon aus, dass sich Linux in der Partition `/dev/hdb13` befindet. Beachten Sie, dass Sie innerhalb von GRUB die Taste (Tab) wie in einer Linux-Shell zur Vervollständigung von Dateinamen verwenden können. Das ist hilfreich, wenn Sie die genauen Namen der Kernel- und Init-RAM-Disk-Dateien nicht auswendig wissen. Je nach Konfiguration kann es sein, dass Sie weitere Kernel-Parameter angeben müssen, z. B. `ide-scsi`.)

```
grub>  root (hd1,12)
  Filesystem type is ext2fs, partition type 0x83
grub>  kernel /boot/vmlinuz-2.4.6-3.1 ro root=/dev/hdb13
  [Linux-bzImage, setup=0x1400, size=0xce7d8]
grub>  initrd /boot/initrd-2.4.6-3.1.img
  [Linx-initrd @ 0xff93000, 0x4ca81 bytes]
grub>  boot
```

Vollständige GRUB-Installation auf einer Diskette

Das obige Beispiel ist zwar für erste GRUB-Experimente geeignet, aber für das tägliche Starten von Linux zu mühsam. Dazu ist die Eingabe der GRUB-Kommandos zu fehleranfällig. Damit auf der Diskette auch ein GRUB-Menü gespeichert werden kann, muss dort ein richtiges Dateisystem eingerichtet werden. (Genau dieser Schritt wurde oben mit den `dd`-Kommandos vermieden.)

Es gibt mehrere Möglichkeiten, um eine richtige GRUB-Installation durchzuführen: Sie können das Kommando `grub-install` zu Hilfe nehmen oder Sie können `grub` starten und die Installation mit den GRUB-Kommandos `setup` oder `install` durchführen. An dieser Stelle wird ausschließlich die letzte (und leider auch komplizierteste) Variante beschrieben, weil diese zurzeit als einzige die Möglichkeit bietet, GRUB mit einem eigenen Menü zu installieren.

Mit den folgenden Kommandos legen Sie auf einer Diskette ein ext2-Dateisystem an, binden das Dateisystem mit dem Pfad `/floppy` in den Verzeichnisbaum ein, erstellen das

Verzeichnis `/floppy/myboot` und kopieren alle erforderlichen Dateien dorthin: die diversen `*stage*`-Dateien aus dem GRUB-Verzeichnis auf der Festplatte sowie das Programm `grub` selbst.

```
root#  mke2fs /dev/fd0
root#  mkdir /floppy
root#  mount -t ext2 /dev/fd0 /floppy
root#  mkdir /floppy/myboot
root#  cd /floppy/myboot
root#  cp /usr/share/grub/i386-pc/* .
root#  cp /sbin/grub .
```

Der nächste Schritt besteht darin, eine minimale GRUB-Menüdatei zu erstellen. Dazu können Sie jeden beliebigen Editor verwenden. Die folgenden Zeilen können nur als Muster dienen. `root` gibt die Partition an, auf der sich der Linux-Kernel befindet (für dieses Beispiel wäre das `/dev/hdb13`). Die Schlüsselwörter `kernel` und `initrd` geben den Ort der Kernel- und Init-RAM-Disk-Dateien an, wobei die Pfadangabe relativ zum `root`-Device ist. Beachten Sie, dass für die Kernel-Parameter (und insbesondere für die Angabe der Kernel-`root`-Devices) die Linux-Nomenklatur zur Anwendung kommt. Deswegen heißt es hier korrekt `root=/dev/hdb13`!

```
# /floppy/myboot/grub.conf (GRUB menu)
timeout=30
title Linux
  root (hd1,12)
  kernel /boot/vmlinuz-2.4.6-3.1 ro root=/dev/hdb13 hdd=ide-scsi
  initrd /boot/initrd-2.4.6-3.1.img
```

Jetzt müssen Sie nur noch GRUB in den Bootsektor der Diskette installieren. Dazu starten Sie das Programm `grub`:

```
root#  cd /floppy/myboot
root#  grub
```

Im GRUB-Shell-Modus führen Sie das folgende `install`-Kommando aus. Beachten Sie, dass das gesamte Kommando ohne \-Zeichen in einer Zeile angegeben werden muss:

```
grub>  install (fd0)/myboot/stage1 (fd0) (fd0)/myboot/stage2 \
         p (fd0)/myboot/grub.conf
grub>  quit
```

Dieses vollkommen kryptische Kommando bedarf natürlich einer Erklärung. In allgemeiner Form lautet die Syntax:

```
install  stage1 [d]  installdevice stage2 p  config
```

`stage1` und `stage2` sind zwei unterschiedliche Programmteile, die beide für den Start von GRUB erforderlich sind. Der Dateiname muss jeweils in der vollständigen GRUB-Nomenklatur angegeben werden, damit GRUB weiß, von wo es die Dateien lesen soll. Das Installations-Device gibt an, auf welchem Datenträger `stage1` installiert werden soll.

Die Angabe (fd0) bewirkt, dass `stage1` in den Bootsektor der Diskette geschrieben wird. Dem Installations-Device muss der Buchstabe 'd' vorangestellt werden, wenn es sich beim Datenträger für die Installation und beim Datenträger, der die GRUB-Dateien enthält, um unterschiedliche Festplatten oder Disketten handelt. (Das ist hier nicht der Fall, alle Dateien befinden sich auf der Diskette.)

Glücklicherweise müssen Sie das `install`-Kommando nur ein einziges Mal ausführen. Wenn Sie später das Menü verändern möchten, reicht es aus, die Datei `grub.conf` zu verändern. GRUB hat sich den Dateinamen gemerkt und findet die veränderte Datei selbstständig.

Jetzt können Sie die Diskette aus dem Dateisystem lösen und den Rechner mit der Diskette neu starten. Wenn alles klappt, erscheint ein aus nur einem Eintrag bestehendes Menü, mit dem Sie Linux starten können:

```
root#  cd
root#  umount /floppy
```

GRUB-Installation in den MBR der Festplatte

VORSICHT

Die Installation von GRUB in den Master-Bootsektor der Festplatte ist eine gefährliche Operation. Wenn dabei etwas schief geht, können Sie Ihren Rechner nicht mehr starten! Es ist daher unbedingt erforderlich, dass Sie vor der Ausführung der folgenden Kommandos eine Bootdiskette erstellen (und testen!), mit der Sie Linux starten können, und dass Sie vom MBR der Festplatte eine Sicherungskopie erstellen (siehe Seite 329), damit Sie diesen zur Not wiederherstellen können. Darüber hinaus ist es eine gute Idee, die nicht ganz unkomplizierte Installation von GRUB zuerst mit einer Diskette üben (siehe oben).

Grundsätzlich erfolgt die GRUB-Installation in den MBR (*Master Boot Record*) der Festplatten wie die GRUB-Installation auf eine Diskette. Dazu kopieren Sie die GRUB-Dateien üblicherweise in das Verzeichnis `/boot/grub`, sofern sie sich noch nicht dort befinden:

```
root#  mkdir /boot/grub
root#  cp /usr/share/grub/i386-pc/* /boot/grub
```

Anschließend erstellen Sie die GRUB-Menüdatei, deren Aufbau sich nicht von der Diskettenversion unterscheidet. (Weitere Hinweise zu dieser Datei finden Sie im vorherigen Abschnitt zur GRUB-Installation auf eine Diskette.)

```
# /boot/grub/grub.conf (GRUB menu)
timeout=30
title Linux
  root (hd1,12)
  kernel /boot/vmlinuz-2.4.6-3.1 ro root=/dev/hdb13 hdd=ide-scsi
  initrd /boot/initrd-2.4.6-3.1.img
```

Zu guter Letzt starten Sie `grub` und führen dort das folgende `install`-Kommando aus. Beachten Sie, dass das gesamte Kommando ohne \-Zeichen in einer Zeile angegeben werden muss. Statt `(hd1,12)` müssen Sie den GRUB-Device-Namen Ihrer Festplattenpartition angeben, auf der sich das `/boot`-Verzeichnis befindet. `(hd0)` bezeichnet den Ort, wohin GRUB installiert werden soll (also die erste Festplatte). Neu ist der Buchstabe 'd', der `(hd0)` vorangestellt wird. Dieser Buchstabe ist immer dann erforderlich, wenn der Installationsort und der Ort der GRUB-Dateien unterschiedlich ist:

```
grub>  install (hd1,12)/boot/grub/stage1 \
               d (hd0) \
               (hd1,12)/boot/grub/stage2 \
               p (hd1,12)/boot/grub/grub.conf
grub>  quit
```

Aufbau der GRUB-Menüdatei

Dieser Abschnitt beschreibt die wichtigsten Elemente einer GRUB-Menüdatei. Beachten Sie, dass Sie fast alle hier vorgestellten Kommandos auch interaktiv in GRUB ausführen können.

TIPP

Wenn Sie eine GRUB-Menüdatei rasch testen möchten, starten Sie unter Linux `grub` und führen dort das folgene Kommando aus:

```
configfile (hd1,12)/boot/grub/grub.conf
```

Statt `(hd1,12)` müssen Sie den GRUB-Namen für die Festplattenpartition angeben, in der sich die GRUB-Menüdatei befindet. Wenn alles klappt, zeigt GRUB das Menü an. Sie können allerdings kein Betriebssystem tatsächlich starten (weil Linux ja schon läuft).

Aufbau: Grundsätzlich besteht die GRUB-Menüdatei aus einem globalen Bereich, der diverse Defaulteinstellungen enthält, sowie mehreren Menüeinträgen, die jeweils mit der Zeile `title` beginnen. Der nach `title` angegebene Text ist der Inhalt der Menüzeile. Die weiteren Zeilen (bis zur nächsten `title`-Anweisung bzw. bis zum Ende der Datei) sind GRUB-Kommandos, die in dieser Reihenfolge ausgeführt werden. (Wenn Sie die Kommandos interaktiv testen, müssen Sie zusätzlich noch `boot` ausführen. Dieses Kommando muss in der Menüdatei nicht angegeben werden.)

Globaler Bereich: Die folgenden Zeilen erläutern einige Kommandos, die im globalen Bereich der GRUB-Menüdatei vorkommen können. Mit `splashimage` kann eine Hintergrundgrafik angegeben werden. Die Grafik muss 640x480 Pixel groß sein, im XPM-Format (8 Bit pro Pixel) vorliegen und mit `gzip` komprimiert sein.

```
# /boot/grub/grub.conf (globaler Bereich)
default 2    # der dritte Menüeintrag gilt als Default
timeout 30   # 30 Sekunden warten, bevor das
             # Defaultsystem gestartet wird
splashimage=(hd1,12)/boot/grub/splash.xpm.gz
```

Linux starten: Um Linux zu starten, müssen Sie mit `root` die Partition angeben, auf der sich der Linux-Kernel und (falls erforderlich) die Initial-RAM-Disk-Datei befinden. Diese Partition wird für GRUB zur aktiven Partition. Die `kernel`- und `initrd`-Kommandos geben den genauen Ort der Dateien sowie eventuelle Kernel-Optionen an. (Die Angaben beziehen sich auf die durch `root` eingestellte Partition.)

```
# /boot/grub/grub.conf (Linux starten)
title Linux
  root (hd1,12)
  kernel /boot/vmlinuz-2.4.6-3.1 ro root=/dev/hdb13 hdd=ide-scsi
  initrd /boot/initrd-2.4.6-3.1.img
```

Windows starten: Wenn Sie Windows starten möchten, müssen Sie die aktive Partition mit `rootnoverify` statt mit `root` angeben. Das Kommando `chainloader +1` bewirkt, dass der erste Sektor dieser Partition gelesen und ausgeführt wird.

```
# /boot/grub/grub.conf (Windows starten)
title Windows
  rootnoverify (hd0,0)
  chainloader +1
```

Passwortschutz

Die große Flexibilität, die GRUB bietet, ist natürlich zugleich auch eine Gefahrenquelle. Beispielsweise kann der GRUB-Benutzer mit dem Kommando `cat` alle Dateien lesen, die in ext2-, reiserfs- oder vfat-Partitionen gespeichert sind. Deswegen ist es sinnvoll, eine GRUB-Installation durch ein Passwort abzusichern.

Die einfachste Möglichkeit besteht darin, in den globalen Teil der Menüdatei die folgende Zeile einzufügen:

```
# /boot/grub/grub.conf (globaler Bereich)
password abcde
```

Das bewirkt, dass die GRUB-Anwender zwar alle Menükommandos auswählen und ausführen dürfen, dass aber alle anderen interaktiven Möglichkeiten von GRUB gesperrt sind. Wenn Anwender Kernel-Optionen verändern oder andere Kommandos ausführen möchten, müssen sie vorher (P) und dann das Passwort angeben.

Möglicherweise wollen Sie auch einzelne Menüeinträge mit einem Passwortschutz ausstatten. Dazu fügen Sie unmittelbar nach der `title`-Zeile das Kommando `lock` ein. Dieser Menüeintrag kann dann erst nach der Passwortangabe verwendet werden. (Wenn Sie möchten, dass GRUB ohne Passwort gar nicht verwendet werden kann, müssen Sie alle Menüeinträge mit `lock` absichern.)

Sicherheitsexperten werden angesichts der Klartextangabe des Passworts wahrscheinlich die Stirn runzeln. Jeder, der sich Zugriff zur GRUB-Menüdatei verschafft, kann das Passwort unverschlüsselt lesen. Um auch diese Sicherheitslücke zu schließen, bietet GRUB ab

Version 0.90 die Möglichkeit, das Passwort in verschlüsselter Form anzugeben. Zur Verschlüsselung des Passworts führen Sie entweder in GRUB das Kommando `md5crypt` oder außerhalb das Kommando `grub-md5-crypt` aus. In beiden Fällen werden Sie nun zur Eingabe Ihres Passworts aufgefordert. Als Ergebnis erhalten Sie eine Zeichenkette mit dem verschlüsselten Passwort.

```
root#  grub-md5-crypt
Password:  xxxxx
2s34jd8d1f24jkSdF8
```

Diese Zeichenkette geben Sie nun in der Menüdatei an, wobei Sie bei `password` die Option `--md5` verwenden.

```
# /boot/grub/grub.conf (globaler Bereich)
password --md5 2s34jd8d1f24jkSdF8
```

Leider konnte ich den GRUB-Passwortschutz mit Verschlüsselung nicht testen, weil bei der von mir eingesetzten GRUB-Version 0.90 entgegen der mitgelieferten Dokumentation das Kommando `grub-md5-crypt` noch nicht funktionierte. In zukünftigen Versionen sollte dieses Problem aber hoffentlich behoben sein.

8.4 Kernel-Bootoptionen

Bei der Konfiguration von LILO können Sie mit `append` Kernel-Optionen angeben. Derartige Optionen können Sie auch interaktiv beim Start eines Linux-Installationsprogramm oder beim Start von LILO über die Tastatur eintippen. Kernel-Optionen helfen oft dabei, Hardware-Probleme zu umgehen. Wenn der Linux-Kernel beispielsweise nicht erkennt, wie viel RAM Ihr Rechner hat (das ist eigentlich ein BIOS-Problem), können Sie den korrekten Wert mit dem Parameter `mem=` angeben.

```
LILO
boot: {\bfs linux option=parameter1,parameter2 option=parameter ...}
```

VERWEIS

Selbstverständlich bietet auch GRUB die Möglichkeit, zusätzliche Kernel-Bootoptionen anzugeben. Die Vorgehensweise ist auf Seite 349 beschrieben.

Die Parameter zu einer Option müssen ohne Leerzeichen angegeben werden (z. B. `ether=10,0x300,0,0,eth0`). Wenn mehrere Optionen angegeben werden, müssen diese durch Leerzeichen (nicht durch Kommata) voneinander getrennt werden.

Hexadezimale Adressen werden in der Form 0x1234 angegeben. Ohne vorangestelltes 0x wird die Zahl dezimal interpretiert. Beachten Sie, dass die Reihenfolge der Parameter nicht konsistent ist. Je nach Option wird manchmal zuerst die I/O-Adresse, manchmal zuerst die IRQ-Nummer angegeben.

HINWEIS

Beachten Sie, dass die beim LILO-Start angegebenen Parameter nur Einfluss auf die in den Kernel integrierten Treiber haben! Parameter für Kernel-Module müssen dagegen in der Datei `/etc/modules.conf` angegeben werden. Detaillierte Informationen zu dieser Datei finden Sie im nächsten Kapitel.

VERWEIS

Dieser Abschnitt beschreibt nur die wichtigsten Kernel-Optionen. Erheblich mehr Informationen erhalten Sie mit `man 7 bootparam`. Falls Sie Zugang zum SuSE-Referenzhandbuch haben, lohnt sich das Studium des Kapitels *Kernel-Parameter*, das sehr praxisnahe Informationen enthält. Falls der Linux-Kernel-Code installiert ist, sollten Sie einen Blick in die folgenden Dateien werfen:

```
/usr/src/linux/Documentation/kernel-parameters.txt
/usr/src/linux/Documentation/ide.txt
/usr/src/linux/Documentation/*
/usr/src/drivers/*/README.*
```

Wichtige Kernel-Optionen

`root=/dev/hdb3`
Gibt an, dass nach dem Laden des Kernels die dritte primäre Partition des zweiten IDE-Laufwerks als Root-Dateisystem verwendet werden soll. (Analog können natürlich auch andere Laufwerke (auch SCSI) und Partitionen angegeben werden.) Die Option ist dann sinnvoll, wenn ein bereits installiertes Linux-System über eine Installationsdiskette gebootet werden soll (beispielsweise nachdem durch die Windows-9x-Installation der Bootsektor für LILO zerstört worden ist).

`ro`
Gibt an, dass das Dateisystem read-only gemountet werden soll. Das ist (in Kombination mit einer der beiden folgenden Optionen) praktisch, wenn ein defektes Dateisystem manuell repariert werden muss.

`init`
Nach dem Kernel-Start wird automatisch das Programm `/sbin/init` ausgeführt (Init-V-Prozess, siehe Seite 359). Wenn Sie dies nicht wollen, können Sie mit der Option `init` ein anderes Programm angeben. Mit `init=/bin/sh` erreichen Sie beispielsweise, dass eine Shell gestartet wird. Die Option kann Linux-Profis helfen, ein Linux-System wieder zum Laufen zu bringen, wenn bei der Init-V-Konfiguration etwas schief gegangen ist. Beachten Sie, dass das root-Dateisystem nur read-only zur Verfügung steht (siehe Seite 267), dass in der Konsole das US-Tastatur-Layout gilt, dass die `PATH`-Variable noch leer ist etc.

`single`
`emergency`
Wenn eine der zwei obigen Optionen verwendet wird, startet der Rechner im Single-User-Modus. (Genau genommen werden diese Optionen nicht vom Kernel ausgewertet, sondern so wie alle unbekannten Optionen an das erste vom Kernel gestartete Programm

weitergegeben. Dabei handelt es sich um `/sbin/init`, das für die Initialisierung des Systems zuständig ist (siehe auch Seite 359).

`initrd=`
`noinitrd`
Die beiden obigen Optionen bewirken jeweils, dass die zur Verfügung stehende Initial-RAM-Disk *nicht* geladen wird. Das ist dann sinnvoll, wenn Sie in dieser Datei einen Fehler vermuten oder wenn Sie eine Installations-CD-ROM zum Start eines schon vorhandenen Linux-Systems verwenden möchten. (Bei einer Installations-CD-ROM enthält die Initial-RAM-Disk unter anderem das Installationsprogramm, das Sie aber nicht ausführen möchten.)

`mem=128M`
Gibt an, dass der Rechner mit 128 MByte RAM ausgestattet ist. Die Option kann bei manchen Rechnern mit mehr als 64 MByte RAM erforderlich sein, weil deren BIOS maximal 64 MByte meldet. Die Angabe erfolgt in kByte (wie im Beispiel oben) oder in MByte (mit einem großen `M`). Achtung: Bei manchen Systemen wird der oberste Bereich des RAMs dazu verwendet, um das BIOS dorthin zu kopieren. Daher kann es sein, dass bei einem System mit 96 MByte RAM tatsächlich etwas weniger Speicher zur Verfügung steht.

`maxcpus=1`
Wenn Sie bei einem Multiprozessorsystem Bootprobleme haben, können Sie mit dieser Option die Anzahl der genutzten Prozessoren auf 1 reduzieren.

`reserve=0x300,0x20`
Gibt an, dass die 32 Bytes (hexadezimal 0x20) zwischen 0x300 und 0x31F von keinem Hardware-Treiber angesprochen werden dürfen, um darin nach irgendwelchen Komponenten zu suchen. Die Option ist bei manchen Komponenten notwendig, die auf solche Tests allergisch reagieren. Die Option tritt im Regelfall in Kombination mit einer zweiten Option auf, die die exakte Adresse der Komponente angibt, die diesen Speicherbereich für sich beansprucht.

`NOPCMCIA=1`
Nur bei SuSE bewirkt die Zuweisung eines beliebigen Werts an die Option `NOPCMCIA`, dass während des Init-V-Prozesses auf den Start des `pcmcia`-Scripts verzichtet wird. (Damit ist die automatische Erkennung von PCMCIA-Karten abgeschaltet.)

IDE-Festplatten: Parameter für Festplatten werden in der Form `hd`x`=`$option$ angegeben. `hda` steht für die erste Festplatte, `hdb` für die zweite usw. (Genau genommen bezeichnen die Buchstaben a, b, c, d etc. das erste Gerät am ersten Controller, das zweite Gerät am ersten Controller, das erste/zweite Gerät am zweiten Controller etc. Je nachdem, ob und wie das CD-ROM-Laufwerk angeschlossen ist, kann die zweite Festplatte ohne weiteres auch `hdc` heißen.)

Parameter für den oder die Festplatten-Controller werden mit `ide`n`=` angegeben; hier kann für n ein Wert zwischen 0 und 3 verwendet werden. (Ein Controller steuert jeweils zwei Platten, also `ide1` für `hdc` und `hdd`. Die meisten IDE-Controller entsprechen zwei

herkömmlichen Controllern und können daher vier Platten steuern.) Es dürfen mehrere Optionen hintereinander angegeben werden.

`hdx=noprobe`
Keinen Test durchführen, ob die Platte existiert bzw. wie groß sie ist. (Die Platte kann dennoch verwendet werden, wenn die Geometrie durch die folgende Option explizit angegeben wird.)

`hdx=1050,32,64`
Die Festplatte hat 1050 Zylinder, 32 Köpfe und 64 Sektoren. Diese Angaben sind nur erforderlich, wenn Linux die Geometrie der Festplatte nicht selbst erkennt.

`hdd=ide-scsi`
Mit dieser Option wird das IDE-Laufwerk als SCSI-Gerät gekennzeichnet. Das ist bei CD-R-Laufwerken sinnvoll (siehe Seite 1010).

TIPP

Sie können die unter DOS gültige Geometrie auch mit dem Programm `DPARAM.COM` ermitteln. Das Programm wird unter DOS mit dem Parameter 0x80 (für die erste Platte) oder 0x81 (zweite Platte) gestartet und liefert als Ergebnis die Anzahl der Zylinder, Köpfe und Sektoren. Das Programm befindet sich auf vielen Linux-CDs im Verzeichnis `dosutils`.

`ide=nodma`
Mit dieser Option wird der DMA-Modus für den Zugriff auf IDE-Geräte deaktiviert. Das ist hilfreich, wenn es Probleme beim Festplattenzugriff gibt und der DMA-Modus per Default aktiv ist (z. B. bei Red Hat 7.1).

8.5 Init-V-Prozess

Dieser Abschnitt beschreibt die Vorgänge, die zwischen dem Einschalten Ihres Rechners und der Login-Aufforderung stattfinden. Nach dem Start des Kernels kann dieser vorläufig nur im Read-Only-Modus auf die Root-Partition zugreifen. Als erstes Programm startet der Kernel das Programm `/sbin/init`. Das Programm `init` kümmert sich dann um die Basiskonfiguration des Systems (Einbinden von Dateisystemen) und den Start zahlloser Dämonen (mit Netzwerkfunktionen, zur Druckerunterstützung usw.).

Die Details der Ausführung des Programms `init` hängen von der jeweiligen Distribution ab. Zwar orientieren sich fast alle Distributionen am System-V-Init-Prozess, wie er auf vielen anderen Unix-Rechnern üblich ist. Dennoch gibt es Unterschiede: insbesondere in welchen Verzeichnissen sich welche Init-Dateien befinden, mit welchen Nummern oder Buchstaben die so genannten Runlevel bezeichnet sind, welche Konfigurationsdateien berücksichtigt werden etc.

Dieser Abschnitt gibt zuerst einen allgemeinen Überblick über den Init-V-Prozess. Am Ende des Abschnitts finden Sie einige Ergänzungen zu den Varianten von Mandrake, Red

Hat und SuSE. Wenn Sie eine andere Distribution verwenden, müssen Sie die Details des Init-V-Prozesses selbst erforschen oder in der Dokumentation nachlesen.

VORSICHT

Versuchen Sie nie, ein Init-V-Paket einer anderen Distribution zu installieren! Auch bei der manuellen Veränderung der Dateien, die den Init-Prozess betreffen, ist allergrößte Vorsicht geboten! Wenn Sie Pech haben, können Sie nach einer inkorrekten Änderung Linux nicht mehr hochfahren. Damit fehlt also auch die Möglichkeit, Ihren Fehler zu korrigieren! (Abhilfe: Versuchen Sie als Erstes, im Single-User-Modus zu starten, indem Sie im LILO als zusätzlichen Bootparameter `single` oder `emergency` angeben. Hilft auch das nichts, müssen Sie mit einer Installationsdiskette booten, das Root-System von Hand mounten und die Reparatur so durchführen. Auf den meisten Installationsdisketten fehlt allerdings ein (brauchbarer) Editor – sorgen Sie also dafür, dass Sie zumindest Backup-Dateien aller veränderten Dateien besitzen.)

VERWEIS

Im folgenden Text ist ständig von Script-Dateien die Rede. Ein Script ist ein meist kleines Programm, das von einem Interpreter ausgeführt wird. Bei allen in diesem Abschnitt behandelten Script-Programmen dient die `bash` als Interpreter. Die `bash` wird unter Linux primär als Kommandointerpreter verwendet (vergleichbar mit `COMMAND.COM` unter DOS). Diese interaktive Verwendung der `bash` ist in Kapitel 20 beschrieben. Eine Einführung in die `bash`-Programmierung finden Sie in Kapitel 21.

Init-V-Überblick

Damit Sie nicht beim Lesen in den Details verloren gehen, folgt zunächst ein kurzer Überblick über einen normalen Linux-Systemstart:

- LILO oder ein anderer Boot-Loader lädt und startet den Kernel.
- Der Kernel startet das Programm `/sbin/init`.
- `init` wertet die Konfigurationsdatei `/etc/inittab` aus.
- `init` führt ein Script zur Systeminitialisierung aus.
- `init` führt das Script `/etc/rc.d/rc` aus.
- `rc` startet diverse Script-Dateien, die sich im Verzeichnis `/etc/rc.d/rc`n`.d` befinden. (n ist der Runlevel – siehe unten.)
- Die Script-Dateien im Verzeichnis `/etc/rc.d/rc`n`.d` starten verschiedene Systemdienste (insbesondere alle Dämonen für die Netzwerkfunktionen).

Runlevel

Der Kernel startet `/sbin/init` als erstes Programm. Dabei werden alle nicht ausgewerteten Kernel-Parameter weitergegeben. Auf diese Weise kann beispielsweise erreicht werden, dass Linux im Single-User-Modus gestartet wird. (Details siehe in der `man`-Page zu `init`.)

`init` ist also der erste laufende Prozess. Alle weiteren Prozesse werden entweder direkt von `init` oder indirekt durch Subprozesse von `init` gestartet. (Wenn Sie `pstree` ausführen, erkennen Sie sofort die dominierende Rolle von `init` – siehe Seite 301.) Beim Herunterfahren des Rechners ist `init` der letzte noch laufende Prozess, der sich um das korrekte Beenden aller anderen Prozesse kümmert.

Für das Verständnis der System-V-Mechanismen ist der Begriff des Runlevels von zentraler Bedeutung: Sie können Ihren Rechner in unterschiedlichen Runleveln betreiben:

- Runlevel 1: Single-User
- Runlevel 2: Multi-User ohne Netzwerk
- Runlevel 3: Multi-User mit Netzwerk, aber ohne automatischen X-Start
- Runlevel 5: Multi-User mit Netzwerk und automatischen X-Start

Für den Shutdown sind die Runlevel 0 und 6 reserviert:

- Runlevel 1: Shutdown mit Halt
- Runlevel 6: Shutdown mit Reboot

Der Runlevel 4 hat üblicherweise keine Funktion.

VORSICHT

Die Runlevel-Nummerierung ist zwischen den Distributionen leider uneinheitlich. Die obige Liste gilt für aktuelle Mandrake-, Red-Hat- und SuSE-Distributionen. Debian verwendet andere Run-Level. Die jeweilige Bedeutung der Runlevel ist immer in `/etc/inittab` dokumentiert.

Default-Runlevel: Der Default-Runlevel wird durch die `initdefault`-Zeile in `/etc/inittab` bestimmt. Bei den meisten aktuellen Distributionen gilt 5 als Default-Runlevel.

TIPP

Der Default-Runlevel bestimmt, ob der Login nach dem Start des Rechners im Textmodus oder unter X erscheint. Um zwischen Text- und X-Login umzuschalten, ändern Sie einfach n in der Zeile `id:`n`:initdefault:`.

`root` kann den Runlevel auch im laufenden Betrieb durch das Kommando `init x` verändern. (Für x muss eine Zahl oder ein Buchstabe angegeben werden, die bzw. der den Runlevel beschreibt.) Beispielsweise ist es für manche Wartungsarbeiten sinnvoll, in den Single-User-Modus zu wechseln. Auch durch `shutdown` und Strg+Alt+Entf wird der Runlevel verändert.

Inittab

Beim Systemstart wird `init` durch die Datei `/etc/inittab` gesteuert. Für die Syntax der `inittab`-Einträge gilt folgendes Schema:

```
id-code:runlevel:action:command
```

`id-code` besteht aus zwei Zeichen, die die Zeile eindeutig identifizieren. Der `runlevel` gibt an, für welchen Runlevel der Eintrag gilt. `action` enthält eine Anweisung für `init`. `command` gibt an, welches Linux-Kommando oder Programm gestartet werden soll.

Die folgende Liste zählt die wichtigsten `action`-Schlüsselwörter auf (eine vollständige Beschreibung erhalten Sie mit `man inittab`):

inittab-Schlüsselwörter

`ctrlaltdel`:	gibt an, wie `init` auf Strg+Alt+Entf reagieren soll.
`initdefault`:	definiert den Default-Runlevel für `init` (siehe oben).
`once`:	`init` startet das angegebene Kommando beim Runlevel-Wechsel.
`respawn`:	`init` startet das Kommando nach seinem Ende wieder neu.
`sysinit`:	`init` startet das Kommando einmal während des Bootprozesses.
`wait`:	`init` wartet auf das Ende des nachfolgenden Kommandos.

Das folgende Listing gibt die leicht gekürzte `inittab`-Datei von Red Hat wieder. Als Default-Runlevel gilt 5. Bei einem normalen Systemstart führt `init` die Script-Dateien `rc.sysinit` und das Kommando `rc 5` aus. Außerdem wird der `update`-Dämon gestartet. (Bei manchen Distributionen heißt dieses Programm `bdflush`. Unabhängig vom Namen ist es dafür zuständig, dass gepufferte Datenblöcke regelmäßig auf der Festplatte gesichert werden.) Schließlich wird für die Textkonsolen 1 bis 6 das Programm `mingetty` gestartet, das einen Login ermöglicht. (Wenn Sie mehr Textkonsolen haben möchten, ist hier der richtige Ort für Veränderungen.)

```
# /etc/inittab für Red Hat 7.2
#   0 - Halt                  4 - unbenutzt
#   1 - Single-User           5 - X11
#   2 - Multi-User ohne NFS   6 - Reboot
#   3 - Multi-User mit NFS
# Default-Runlevel
id:5:initdefault:

# Systeminitialisierung nach einem Reboot
# (Dateisystem testen, eventuell fsck ausführen, Swapping aktivieren,
# alle in fstab genannten Dateisysteme einbinden)
si::sysinit:/etc/rc.d/rc.sysinit

# Start des jeweiligen Runlevel
# (Start von Netzwerkdiensten und anderen Systemprozessen (Dämonen))
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

```
# bei jedem Runlevel starten
# (der update-Dämon kümmert sich um das regelmäßige Speichern
# gepufferter Datenblöcke auf der Festplatte)
ud::once:/sbin/update

# Reaktion auf <Strg>+<Alt>+<Entf>:
# (in 3 Sekunden Shutdown einleiten, Rechner dann neu starten)
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# gettys (Terminal-Emulatoren) für die Textkonsolen starten
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# nur in Runlevel 5: xdm/kdm/gdm starten (X-Login-Manager)
x:5:respawn:/etc/X11/prefdm -nodaemon
```

TIPP

Beim Start von `mingetty` wird der Bildschirm gelöscht. Dagegen wäre an sich nichts einzuwenden, allerdings gehen damit auch die letzten Meldungen des Startprozesses verloren, die manchmal durchaus wichtig sind. Sie können dieses Verhalten vermeiden, wenn Sie bei `mingetty` für die erste Konsole die Option `--noclear` verwenden, also:

```
1:12345:respawn:/sbin/mingetty --noclear tty1
```

Systeminitialisierung

Noch bevor die im nächsten Abschnitt beschriebenen rc-Dateien die runlevel-spezifischen Dienste starten oder stoppen, wird unmittelbar nach dem Rechnerstart eine Systeminitialisierung durchgeführt (`si:`-Zeile in `inittab`). Der Name des Scripts hängt von der Distribution ab:

Mandrake, Red Hat: `/etc/rc.d/rc.sysinit`
SuSE: `/etc/init.d/boot`

Während der Systeminitialisierung werden unter anderem folgende Dinge erledigt:

- diverse Systemvariablen initialisieren (inklusive Host- und Domainname)
- `/proc`-Dateisystem aktivieren
- Datum und Uhrzeit einstellen
- eventuell RAID und LVM aktivieren
- Tastaturlayout für die Textkonsole einstellen
- Swap-Partitionen einbinden

- Dateisystem der root-Partition (und eventuell auch die von weiteren Partitionen) überprüfen (`fsck`)
- ISA-Plug&Play-Hardware-Komponenten initialisieren: `isapnp` konfiguriert Plug&Play-Komponenten auf der Basis der Datei `/etc/isapnp.conf`. Diese Datei kann durch `pnpdump` erzeugt werden und muss dann manuell editiert werden. Falls Sie ISA-Plug&Play-Karten verwenden, sollten Sie einen Blick in die `man`-Texte von `isapnp`, `pnpdump` und `isapnp.conf` werfen.
- Root-Partition im Read-Write-Modus neu einbinden
- eventuell Quota für die Root-Partition aktivieren
- Kernel-Module laden (`/etc/modules.conf`)
- Partitionen einbinden (keine NFS-Partitionen, weil Netzwerkdienste noch nicht zur Verfügung stehen)

Beachten Sie, dass nicht alle der hier beschriebenen Aufgaben direkt vom Systeminitialisierungsscript durchgeführt werden. Zum Teil werden auch andere Script-Dateien eingelesen. Dabei ist die Schreibweise `. name` üblich. (Der Punkt bewirkt, dass die angegebene Datei an dieser Stelle gelesen und ausgeführt wird. Anschließend wird das Script fortgesetzt.)

rc-Dateien

Während der Systeminitialisierung werden nur die Dinge erledigt, die während des Rechnerstarts nur einmal getan werden (z. B. das Einbinden der Partitionen). Alle Systemprozesse, die nur bei Bedarf (je nach Runlevel) gestartet werden sollen und die vielleicht später (bei einem Runlevel-Wechsel) wieder beendet werden sollen, werden dagegen in zahllosen `rc`-Script-Dateien gestartet bzw. wieder gestoppt.

Dazu wird von `init` das Script `/etc/rc.d/rc` ausgeführt. An `rc` wird der gewünschte Runlevel n übergeben. `rc` führt zuerst einige Initialisierungsarbeiten durch. Dann werden alle `rc`n`.d/K*`-Script-Dateien zum Beenden laufender Prozesse ausgeführt. Schließlich werden alle `rc`n`.d/S*`-Script-Dateien zum Starten der neuen Prozesse für den jeweiligen Runlevel ausgeführt.

Der wesentliche Vorteil dieses auf ersten Blick unübersichtlichen Systems besteht darin, dass es sehr einfach ist, neue Systemprozesse in den Init-V-Prozess einzubauen: Es müssen lediglich die rc-Start- und Stopp-Scripts in die richtigen Verzeichnisse kopiert werden. (Genau das geschieht, wenn Sie mit `rpm` einen neuen Netzwerk-Server installieren.) Die folgende Aufstellung (Red Hat) zeigt, welche Script-Dateien für einige ausgewählte Runlevel ausgeführt werden.

```
rc1.d  (Single-User)
K03rhnsd     K20pppoe       K60crond       K86nfslock     K95kudzu
K05anacron   K20rwhod       K60lpd         K87portmap     S00single
K10xfs       K25sshd        K65identd      K88syslog      S17keytable
K15gpm       K30sendmail    K74apmd        K90mysql
K15httpd     K44rawdevices  K74nscd        K90network
K20isdn      K50xinetd      K75netfs       K92ipchains
K20nfs       K60atd         K80random      K92iptables
```

```
rc5.d  (Multi-User mit Netzwerk und X)
K03rhnsd     S08iptables    S25netfs       S80isdn        S90xfs
K20nfs       S10network     S26apmd        S80pppoe       S95anacron
K20rwhod     S12syslog      S40atd         S80sendmail    S99local
K65identd    S13portmap     S55sshd        S85gpm
K74nscd      S14nfslock     S56rawdevices  S85httpd
S05kudzu     S17keytable    S56xinetd      S90crond
S08ipchains  S20random      S60lpd         S90mysql

rc6.d   (Reboot)
K03rhnsd     K20nfs         K60atd         K80random      K92iptables
K05anacron   K20pppoe       K60crond       K86nfslock     K95kudzu
K05keytable  K20rwhod       K60lpd         K87portmap     S00killall
K10xfs       K25sshd        K65identd      K88syslog      S01reboot
K15gpm       K30sendmail    K74apmd        K90mysql
K15httpd     K44rawdevices  K74nscd        K90network
K20isdn      K50xinetd      K75netfs       K92ipchains
```

Genau genommen befinden sich in den `rc`n`.d`-Verzeichnissen nicht unmittelbar die Script-Dateien, sondern lediglich Links darauf. Das hat den Vorteil, dass die Script-Dateien für mehrere Runlevel verwendet und zentral verändert werden können. Die eigentlichen Script-Dateien sind im Verzeichnis `/etc/rc.d/init.d` gespeichert (SuSE: `/etc/init.d`).

Die Namen der Links sind keineswegs so willkürlich, wie sie aussehen: Der Anfangsbuchstabe gibt an, ob es sich um ein Start- oder Kill-Script handelt. (Die S- und K-Links verweisen auf dieselbe Datei; allerdings wird das Script je nach Anfangsbuchstabe von `rc` mit dem Parameter `start` oder `stop` ausgeführt.) Die nachfolgende Nummer bestimmt die Reihenfolge, in der die Script-Dateien ausgeführt werden. (`sendmail` kann beispielsweise erst dann gestartet werden, wenn die Netzwerk-Software schon läuft.)

Ein gemeinsames Merkmal aller Runlevel-Script-Dateien besteht darin, dass sie wahlweise mit den Parametern `start` oder `stop` ausgeführt werden (je nachdem, ob die Funktion gestartet oder beendet werden soll). Fallweise sind auch die Parameter `status`, `restart` und `reload` vorgesehen. Das ist dann sinnvoll, wenn einzelne Funktionen nach der Veränderung von Konfigurationsdateien manuell neu gestartet werden sollen (also ohne einen Runlevel-Wechsel).

TIPP

Je nach Distribution werden verschiedene Programme mitgeliefert, um den Init-V-Prozess zu konfigurieren, beispielsweise `ksysv` (KDE), `tksysv` (Tcl/Tk), `chkconfig` (Mandrake und Red Hat), `serviceconf` (Red Hat) und `drakxservices` (Mandrake).

VERWEIS

Eine Kurzbeschreibung der wichtigsten Dämonen, die durch `rc`-Script-Dateien gestartet werden, finden Sie auf Seite 309. Dort fehlt allerdings die Beschreibung des `network`-Scripts – und zwar deswegen, weil es sich dabei um keinen Dämon handelt. Das `network`-Script initialisiert vielmehr die Netzwerkfunktionen (Netzwerkkarten, Routing etc.). Detaillierte Informationen zu diesem Script finden Sie auf Seite 605.

HINWEIS

Noch ein Hinweis zu den Parametern `restart` und `reload`, die an die meisten Script-Dateien übergeben werden dürfen:

`reload` bietet sich dann an, wenn geänderte Konfigurationsdateien neu eingelesen werden sollen, ohne den Dämon dabei ganz zu stoppen.

`restart` bewirkt dagegen meistens, dass der Dämon vollkommen gestoppt und anschließend neu gestartet wird. Eventuell vorhandene Verbindungen zu Clients können dabei verloren gehen.

Beachten Sie, dass `reload` bei manchen Dämonen nicht vorgesehen ist oder nicht funktioniert! Verwenden Sie in diesem Fall `restart` oder zur Not zuerst `stop` und dann `start`. Den Parameter `restart` sollten Sie auch verwenden, wenn Sie die Konfigurationsdateien verschoben, kopiert etc. haben (z. B. bei einem Backup) – bei `reload` besteht die Gefahr, dass die Backup-Datei statt der neuen Konfigurationsdatei eingelesen wird!

Red-Hat-Besonderheiten

Interaktiver Init-V-Prozess: Falls während der Systeminitialisierung die Taste (I) gedrückt wurde, wird mit `touch` die Datei `/var/run/confirm` erzeugt; die Existenz dieser Datei wird von `/etc/rc.d/rc` überprüft. Falls sie existiert, erscheint in der Folge bei der Ausführung aller `rc`n`.d`-Script-Dateien eine Rückfrage (YES/NO/CONTINUE, wobei CONTINUE bedeutet, dass die weiteren Script-Dateien ohne Rückfrage ausgeführt werden sollen).

Der interaktive Modus ist praktisch, wenn der Rechner während des Init-V-Prozesses hängen bleiben sollte. In diesem Fall kann das betreffende Init-Script herausgefunden und übersprungen werden. Beachten Sie aber, dass der interaktive Modus erst nach dem Ende der Systeminitialisierung wirksam wird. Tritt bereits dort ein Problem auf, haben Sie darauf keinen Einfluss.

chkconfig: Das Kommando `chkconfig` ermöglicht eine effziente Verwaltung der Runlevel-Scripts. Mit der Option `--list` gibt das Kommando eine Übersicht über alle Scripts und zeigt an, in welchem Runlevel sie gestartet werden.

```
root# chkconfig --list
alsa            0:off   1:off   2:on    3:on    4:on    5:on    6:off
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:off   4:off   5:off   6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
cups            0:off   1:off   2:on    3:on    4:on    5:on    6:off
...
xfs             0:off   1:off   2:on    3:on    4:on    5:on    6:off
xinetd          0:off   1:off   2:off   3:on    4:on    5:on    6:off
```

```
xinetd based services:
        chargen-udp:    off
        chargen:        off
        cups-lpd:       off
        ...
        time:           off
```

Mit `--del` kann der Start eines Runlevel-Scripts generell verhindert werden. (Das Script selbst bleibt erhalten, nur die Links in den `rc`n`.d`-Verzeichnissen werden gelöscht.)

```
root#  chkconfig --del kudzu
```

`chkconfig --add` fügt für alle Runlevel Start- und Stopp-Links für einen neuen Service ein. (Damit das funktioniert, muss in den Kommentarzeilen des Init-V-Scripts angegeben sein, für welche Level das Script per Default gestartet werden soll.)

Die Optionen `--add` und `--del` funktionieren auch für `xinetd`-Dienste – siehe Seite 792.

Wenn ein Script in bestimmten Leveln gestartet bzw. gestoppt werden soll, kann die neue Konfiguration mit `--level` definiert werden. Die folgenden Kommandos veranschaulichen die Syntax:

```
root#  chkconfig --level 012346 xfs off
root#  chkconfig --level 5 xfs on
root#  chkconfig --list xfs
xfs    0:off   1:off   2:off   3:off   4:off   5:on    6:off
```

serviceconf: Beginnend mit Red Hat 7.2 steht zur Init-V-Administration auch das X-Programm `serviceconf` zur Verfügung. Es ermöglicht es, einzelne Dämonen und `xinetd`-Netzwerkdienste manuell zu starten, zu stoppen oder neu zu starten. Außerdem können Sie für einen vorher ausgewählten Runlevel per Mausklick angeben, welche Dienste per Default gestartet werden sollen.

Automatische Hardware-Erkennung (kudzu): Im Rahmen des Init-V-Prozesses versucht das System automatisch, Veränderungen an der eingebauten bzw. angeschlossenen Hardware zu erkennen – siehe Seite 454.

Runlevel-Wechsel: Bei einem Runlevel-Wechsel werden nur solche Funktionen gestoppt, die im vorigen Runlevel gestartet wurden, im neuen Runlevel aber nicht mehr benötigt werden. Ebenso werden nur solche Funktionen neu gestartet, die bisher noch nicht aktiv waren. Um das festzustellen, wird beim Start jedes Systemprozesses eine Datei in `/var/lock/subsys` angelegt. Diese Datei wird beim Ende des Prozesses wieder gelöscht.

Lokale Anpassung: In der Datei `/etc/rc.d/rc.local` können lokale Anpassungen an den Init-V-Prozess durchgeführt werden. Das Script sollte ausschließlich Kommandos enthalten, die nur ein einziges Mal beim Systemstart (nicht aber bei jedem Runlevel-Wechsel) ausgeführt werden sollen. `rc.local` wird nach allen `rc`-Scripts ausgeführt.

VERWEIS

Weitere Informationen über das Konzept des Red-Hat-Init-V-Prozesses finden Sie im Verzeichnis `/usr/share/doc/initscripts`.

Mandrake-Besonderheiten

Mandrake verwendet im Wesentlichen dieselben Init-V-Scripts wie Red Hat, sodass die oben beschriebenen Besonderheiten auch für Mandrake gelten. Es gibt aber auch ein paar Abweichungen, die hier dokumentiert sind.

drakxservices: Mit diesem Programm können sowohl Init-V-Scripts als auch `xinetd`-Dienste aktiviert bzw. deaktiviert werden. Soweit die Einstellungen Init-V-Scripts betreffen, werden die Änderungen gleichermaßen für die Runlevel 2 bis 5 durchgeführt. (Für manche Dienste – etwa für den X-Font-Server `xfs` – ist diese Vorgehensweise allerdings nicht sinnvoll.)

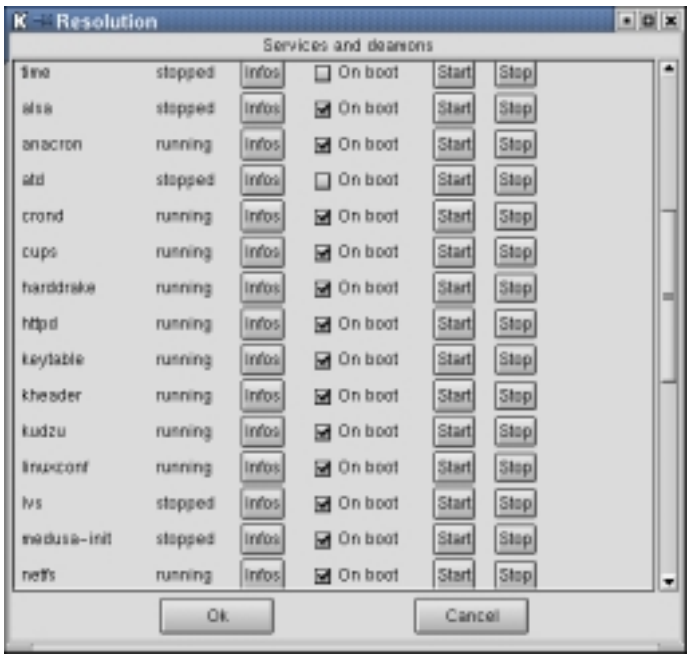

Abbildung 8.1: Steuerung der Init-V-Dämonen und der xinetd-Dienste mit drakxservices

Automatische Hardware-Erkennung (harddrake): Das Programm `harddrake` übernimmt bei Mandrake dieselben Aufgaben wie `kudzu` bei Red Hat – siehe Seite 454.

Grafischer Init-V-Prozess (Aurora): Bei Mandrake erscheinen die Kernel- und Init-V-Meldungen im Grafikmodus. (Das setzt eine VESA-kompatible VGA-Karte voraus.) Der

aktuelle Status des Startprozesses wird durch Icons symbolisiert, die mit der Maus angeklickt werden können. Mandrake verwendet dazu das Programm Aurora:

`http://aurora.mini.dhs.org/`

Aurora wird in `/etc/rc.d/rc` gestartet. Wenn Sie ohne Aurora starten möchten, wählen Sie im LILO-Menü einfach die Variante `nonfb`. Wenn Sie Aurora ganz deaktiveren möchten, entfernen Sie die Option `vga=778` aus `/etc/lilo.conf` und führen Sie `lilo` neuerlich aus. Damit wird der Rechner im Text- statt im Grafikmodus gestartet. Sie können natürlich auch das Paket `Aurora` deinstallieren.

SuSE-Besonderheiten

Init-V-Script-Dateien: Die Dateien befinden sich in `/etc/init.d`. (Bei früheren SuSE-Versionen befanden sie sich in `/sbin/init.d`.)

rc.config: SuSE verwendet `/etc/rc.config` sowie die Dateien im Unterverzeichnis `/etc/rc.config.d` als zentrale Konfigurationsdateien. Diese Konfigurationsdateien werden üblicherweise mit YaST verändert; sie werden von fast allen Scripts des Init-V-Prozesses berücksichtigt.

Die Dateien sind so gut mit Kommentaren versehen, dass in vielen Fällen auch manuelle Änderungen möglich sind. Allerdings müssen Sie anschließend das Programm `SuSEconfig` ausführen, damit die neuen Einstellungen auch auf andere Systemdateien übertragen werden. (Wenn Sie `rc.config` via YaST oder YaST2 verändern, wird `SuSEconfig` automatisch ausgeführt.)

TIPP

Wenn Sie als Systemadministrator bisher mit anderen Distributionen gearbeitet haben, ist `rc.config` sicherlich die wichtigste SuSE-Besonderheit, an die Sie sich gewöhnen müssen. Wenn etwas nach einer manuellen Konfiguration nicht so funktioniert, wie Sie es erwarten bzw. von anderen Distributionen gewohnt waren, ist meistens eine versteckte Einstellung in einer der `rc.config`-Dateien schuld.

Lokale Anpassung: In der Datei `/etc/rc.d/boot.local` können lokale Anpassungen an den Init-V-Prozess durchgeführt werden. Das Script sollte ausschließlich Kommandos enthalten, die nur ein einziges Mal beim Systemstart (nicht aber bei jedem Runlevel-Wechsel) ausgeführt werden sollen. Ein typischen Beispiel sind `modprobe`-Anweisungen, um ein ganz bestimmtes Kernel-Modul zu laden. `rc.local` wird vor den `rc`-Scripts ausgeführt.

Wenn Sie eigene Runlevel-Funktionen in den Init-V-Prozess integrieren möchten, sollten Sie als Vorbild für die Start-/Stoppdatei das Muster `/etc/init.d/skeleton` verwenden. SuSE-konforme Runlevel-Dateien können mit den Parametern `start`, `stop`, `restart`, `reload`, `probe` und `status` ausgeführt werden.

insserv: Um Links von `/etc/rc.d/rc`n`.d` auf ein neues Init-V-Script zu setzen, können Sie einfach das Kommando `insserv scriptname` ausführen. `insserv` wertet die

Kommentare innerhalb des Scripts aus, die angeben, für welche Runlevel das Script ausgeführt werden soll.

Runlevel-Wechsel: Bei einem Runlevel-Wechsel testet `/etc/rc.d/rc` durch einen Vergleich der Verzeichnisse `/etc/rc.d/rc`*neu*`.d` und `rc`*alt*`.d`, welche Funktionen sich durch den Runlevel-Wechsel ändern. Nur solche Funktionen werden gestoppt (Script-Dateien `rc`*alt*`.d/K*`) bzw. neu gestartet (`rc`*neu*`.d/S*`). Funktionen, die durch den neuen Runlevel-Wechsel unverändert bleiben, werden nicht gestoppt und neu gestartet.

Splash-Bildschirm: Seit SuSE 7.2 werden die Meldungen während des Systemstarts auf einer grafisch verzierten Textkonsole ausgegeben, die als Splash-Bildschirm bezeichnet wird. Dazu hat SuSE die entsprechenden Funktionen in den Kernel integriert. Der Splash-Bildschirm erscheint automatisch, wenn der VGA-Modus 771 verwendet wird. Wenn Sie den Splash-Bildschirm deaktiveren möchten, verwenden Sie entweder einen anderen VGA-Modus oder die Kernel-Option `splash=0`.

```
# in /etc/lilo.conf
# SuSE-Splash-Bildschirm deaktiveren
vga = normal
...
image = ...
  append = "splash=0"
```

VERWEIS

Weitere Informationen zum SuSE-Systemstart finden Sie in der Datei `/etc/init.d/README` bzw. in der Manualseite zu `init.d`.

8.6 Logging-Dateien und Kernel-Meldungen

Linux protokolliert in so genannten Logging-Dateien sehr viele Operationen: Kernel-Meldungen, das Ein- und Ausloggen von Benutzern, viele Netzwerkoperationen (z. B. die Herstellung einer Internet-Verbindung) etc. Die Logging-Dateien befinden sich üblicherweise im Verzeichnis `/var/log` (manchmal auch in `/var/adm/log`). Logging-Dateien sind sehr oft ein wichtiges Hilfsmittel zur Fehlersuche.

Für das Logging sind zwei so genannte Dämonen (im Hintergrund laufende Systemprozesse) zuständig, die beide im Rahmen des Init-V-Prozesses gestartet werden:

- `klogd` ist für Kernel-Meldungen zuständig.
- `syslogd` protokolliert die Meldungen diverser Programme. Der Dämon wird durch die Datei `/etc/syslog.conf` gesteuert. In dieser Datei können Sie einstellen, was in welche Datei protokolliert werden soll.

Protokollierung des Startprozesses

Während des Linux-Starts huschen bei älteren Distributionen seitenweise Kernel- und Init-V-Meldungen über den Bildschirm. Bei moderneren Distributionen sieht diese Anzeige zwar grafisch ansprechend aus, dennoch ist es fast unmöglich, die dargestellten Informationen aufzunehmen. Daher ist es wichtig, dass die dargestellten Meldungen protokolliert werden.

Kernel-Meldungen: In der ersten Phase des Startprozesses versucht der Linux-Kernel die Hardware zu erkennen. Die dabei angezeigten Meldungen können Sie im laufenden Betrieb jederzeit mit dem Kommando `dmesg` anzeigen. Im Folgenden sind einige Zeilen für Kernel 2.4 abgedruckt.

```
user$ dmesg
Linux version 2.4.4-4GB (root@Pentium.suse.de) (gcc version 2.95.3
  20010315 (SuSE)) #1 Wed May 16 00:37:55 GMT 2001
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
 BIOS-e820: 000000000009fc00 - 00000000000a0000 (reserved)
 ...
Kernel command line:
  auto BOOT_IMAGE=linux ro root=34b BOOT_FILE=/boot/vmlinuz
Initializing CPU#0
Detected 400.917 MHz processor.
Console: colour dummy device 80x25
Calibrating delay loop... 799.53 BogoMIPS
Memory: 254252k/262132k available (1225k kernel code,
  7492k reserved, 933k data, 112k init, 0k highmem)
...
PCI: PCI BIOS revision 2.10 entry at 0xf0550, last bus=1
PCI: Using configuration type 1
PCI: Probing PCI hardware
...
hda: IBM-DHEA-38451, ATA DISK drive
hdb: Maxtor 33073U4, ATA DISK drive
hdc: IBM-DHEA-38451, ATA DISK drive
hdd: Pioneer DVD-ROM ATAPIModel DVD-103S 011, ATAPI CD/DVD-ROM drive
...
Partition check:
 hda: hda1 hda2 < hda5 hda6 hda7 hda8 hda9 hda10 hda11
                  hda12 hda13 > hda3
 hdb: hdb1 hdb2 < hdb5 hdb6 hdb7 hdb8 hdb9 hdb10 hdb11 >
 hdc: hdc1 hdc2 < hdc5 hdc6 hdc7 hdc8 hdc9 hdc10 hdc11 hdc12 >
...
```

Beachten Sie, dass der durch `dmesg` dargestellte Zwischenspeicher für Kernel-Meldungen begrenzt ist. Wenn dieser Speicher voll ist, werden die ältere Meldungen überschrieben. Da der Kernel auch im laufenden Betrieb Meldungen produziert, werden Sie ein vollständiges Protokoll der Startmeldungen mit `dmesg` nur unmittelbar nach dem Sys-

temstart erhalten. Aus diesem Grund werden die Kernel-Meldungen bei den meisten Distributionen auch in eine Datei geschrieben:

Red Hat, Mandrake: `/var/log/dmesg`
SuSE: `/var/log/boot.msg` (enthält auch Init-V-Meldungen)

Init-V-Meldungen: Sobald der Kernel läuft, beginnt der Init-V-Prozess. Auch dessen Meldungen werden in einer Datei gespeichert. Abermals hängt der Ort von der Distribution ab:

Red Hat, Mandrake: `/var/log/boot.log`
SuSE: `/var/log/boot.msg` (enthält auch Kernel-Bootmeldungen)

Bei Red Hat und Mandrake wird `boot.log` bei jedem Rechnerstart ergänzt. Die Datei enthält daher Init-V-Meldungen von vielen Rechnerstarts und -stopps. Bei SuSE wird `boot.msg` dagegen bei jedem Neustart neu initialisiert. Die Meldungen des letzten Starts finden Sie in `boot.omsg`.

Administration der Logging-Dateien

Logging-Dateien haben die unangenehme Eigenschaft, dass ihre Größe allmählich ins Uferlose wächst. Daher ist es hin und wieder notwendig, mit einer neuen Logging-Datei zu beginnen. Die bisherige Logging-Datei sollte bei dieser Gelegenheit umbenannt und möglichst auch komprimiert werden. Das ergibt dann Dateien wie `messages.1.gz`, `messages.2.gz` etc. Bei den meisten Distributionen kümmert sich ein `cron`-Job um diese Arbeit:

Mandrake/Red Hat: `/etc/cron.daily/logrotate` ruft das Programm `logrotate` auf, das durch die Dateien `/etc/logrotate.conf` und `/etc/logrotate.d/*` gesteuert wird.

Bei Mandrake können Sie zum Lesen der Protokolldateien das Programm `logdrake` verwenden – siehe Seite 1196.

SuSE: `/etc/cron.daily/aaa_base_rotate_logs` enthält selbst den Code zur Bearbeitung der Log-Dateien; die Steuerung erfolgt durch `/etc/rc.config` und `/etc/logfiles`.

TIPP Werfen Sie hin und wieder einen Blick in das `/var/log`-Verzeichnis und löschen Sie alte, nicht mehr benötigte Logging-Dateien!