Linux BootPrompt HOWTO

Paul Gortmaker (p_gortmaker@yahoo.com) und Marco Budde (Budde@tu-harburg.de) v1.2, 20. März 2000

Dies ist das BootPrompt HOWTO, welches eine Zusammenstellung aller möglichen Bootparameter enthält, die während des Bootvorgangs an den Linux-Kernel geschickt werden können. Hierbei sind alle Kernel- und Geräteparameter eingeschlossen. Es wird diskutiert, wie der Kernel Bootparameter sortiert und man erhält einen Überblick über die bekannteste Software, die zum Booten von Linux-Kerneln verwendet wird.

Inhaltsverzeichnis

1 Einführung

Der Kernel verfügt über eine begrenzte Fähigkeit, während des Bootvorgangs Informationen in Form einer Kommandozeile anzunehmen, ähnlich einer Parameterliste, die man einem Programm übergeben würde. Dies dient im allgemeinen dazu, den Kernel mit Informationen über Hardwareparameter zu versorgen, die der Kernel alleine nicht bestimmen könnte oder die Werte zu vermeiden/übergehen, die der Kernel anderenfalls erkennen würde.

Will man jedoch lediglich ein Kernelimage direkt auf Diskette kopieren (z.B. `cp zImage /dev/fd0`), dann erhält man nicht die Möglichkeit, einen Parameter für diesen Kernel festzulegen. Deshalb werden die meisten Linux-Anwender Software wie *LILO* oder *loadlin* verwenden, die diese Parameter an den Kernel weiterleitet und ihn anschließend bootet.

Die derzeitige Ausgabe dieses Textes behandelt Kernel bis einschließlich Version 2.2.9. Es werden ebenfalls einige Eigenschaften dokumentiert, die ausschließlich bei den Entwicklerversionen des Kernels (bis Version 2.3.2) vorkommen.

1.1 Ablehnungshinweise und Copyright

Dieses Dokument ist *kein* Evangelium. Es bietet jedoch möglicherweise die aktuellste Information, die man finden kann. Jeder ist selbst dafür verantwortlich, was mit der eigenen Hardware geschieht. Falls die Hardware in Rauch und Flammen aufgeht, was eigentlich unmöglich ist, übernehme ich dafür keine Verantwortung.

Falls Sie beabsichtigen, dieses Dokument in eine Veröffentlichung aufzunehmen, nehmen Sie bitte Kontakt mit mir auf. Ich werde dann mein Möglichstes tun, Ihnen die aktuellsten Informationen zur Verfügung zu stellen. In der Vergangenheit wurden längst überholte Versionen der Linux-HOWTO-Dokumente veröffentlicht, was den Entwicklern ständigen Kummer bereitete, da sie mit Fragen gequält wurden, die in den neueren Versionen bereits beantwortet waren.

1.2 Für wen ist dieses Dokument gedacht?

Die meisten Anwender werden dieses Dokument nicht benötigen, da Linux seine Aufgabe, die vorhandene Hardware zu erkennen und passende Standardwerte für die meisten Parameter auszuwählen, sehr gut erledigt. Gedacht ist diese HOWTO für diejenigen Anwender, die eventuell einige Standardeinstellung ändern möchten, um den Kernel für den jeweiligen Rechner zu optimieren. Auch Anwender, die sich selbst einen eigenen Kernel »gebacken« haben, um Unterstützung für eine nicht so weit verbreitete Hardware hinzuzufügen, bei der eine automatische Erkennung noch nicht funktioniert, sollten diesen Text gelesen haben.

Wichtiger Hinweis für die Verwendung von Modulen: Ein Kennzeichen von Bootprompt-Parametern ist die ausschließliche Anwendung auf Hardware-Treiber, die direkt in den Kernel einkompiliert sind. Sie haben *keine Auswirkung* auf Treiber, die als Module geladen werden. Die meisten Distributionen verwenden Module. Sollten Zweifel bestehen, sollte man `man depmod` und `man modprobe` zusammen mit `/etc/conf.modules` anschauen.

1.3 Weitere Dokumentationen

Die aktuellsten Dokumentationen werden immer die Kernel-Quelldateien selbst sein. Aber keine Angst, man benötigt keine Programmierkenntnisse zum Lesen der Anmerkungen in den Quelldateien. Will man zum Beispiel wissen, welche Parameter vom AHA1542 SCSI-Treiber erkannt werden, dann geht man ins Verzeichnis `linux/drivers/scsi` und wirft einen Blick auf die Datei `aha1542.c`. Dort findet man bereits in den ersten 100 Zeilen eine einfache Beschreibung (auf Englisch) der Bootparameter, die der 1542-Treiber erkennt.

Das Verzeichnis `linux` findet sich normalerweise unter `/usr/src/`. Bei allen Verweisen auf andere Dateien in diesem Dokument, die Bestandteil der Kernel Sourcen sind, wird deren kompletter Pfadname mit `linux/` abgekürzt. Um den kompletten Pfad zu erhalten, muß `/usr/src` oder was für das jeweilige System passend ist am Anfang hinzugefügt werden. Falls Sie eine Datei garnicht finden können, machen Sie von den Befehlen `find` und `locate` Gebrauch.

Die nächstbeste Informationsquelle sind die Dokumentationsdateien, die mit dem Kernel selbst ausgeliefert werden. Es gibt bereits einige davon und die meisten können im Verzeichnis `linux/Documentation` und seinen Unterverzeichnissen gefunden werden. Es existieren einige `README.foo`-Dateien, die sich in dem jeweiligen Treiber-Verzeichnis befinden (z.B. `linux/drivers/XXX/`, wobei `XXX` `scsi`, `char` oder `net` ist).

Hat man sich für bestimmte Bootparameter entschieden und will nun herausfinden, wie man die Information an den Kernel weitergibt, sollte man einen Blick auf die Dokumentation der Software, wie z.B. LILO oder loadlin werfen, die zum Booten des Kernels verwendet wird. Nachfolgend wird ein kurzer Überblick gegeben, der jedoch keinen Ersatz für die mit der Bootsoftware ausgelieferte Dokumentation darstellt.

1.4 Linux Newsgruppen

Bei Fragen über die Weitergabe von Bootparametern an den Kernel sollte *zuerst* dieses Dokument gelesen werden. Falls dieses Dokument und die oben genannte Dokumentation die Fragen nicht klären können, dann kann man sich an die Linux-Newsgruppen wenden.

Allgemeine Fragen über die Systemkonfiguration sollte man an die Newsgruppe

```
de.comp.os.unix.linux.misc
```

richten. Wir *bitten* darum, sich an die allgemeinen Richtlinien zu halten und vor allem eine Frage nicht gleichzeitig in mehreren Gruppen zu stellen.

Natürlich sollte man zuerst die älteren Artikel der Newsgruppe lesen, anstatt blindlings eine Frage zu stellen. Es ist nämlich gut möglich, daß dieselbe Frage bereits gestellt wurde oder möglicherweise sogar zu den Frequently Asked Questions (häufig gestellte Fragen) gehört. Ein schneller Blick auf die Linux-FAQs ist eine *gute* Idee. Die FAQ sollte man in der Nähe der Ursprungsseite dieses Dokuments finden.

2 Übersicht über die BootPrompt-Parameter

In diesem Abschnitt werden einige Programme vorgestellt, die zur Weiterleitung von Kernel-Bootparametern zum Kernel selbst verwendet werden können. Es wird ebenfalls erklärt, wie die Parameter verarbeitet werden, welchen Beschränkungen sie unterliegen und wie sie zu jedem passenden Gerät, für das sie bestimmt sind, weitergeleitet werden.

Man sollte *unbedingt* beachten, daß innerhalb eines Bootparameters *keine* Leerstellen verwendet werden sollten, nur zwischen getrennten Parametern. Mehrere Werte für einen Parameter müssen durch ein Komma getrennt werden und zwar wiederum ohne Leerstellen. Richtig wäre z.B. dieses Beispiel:

```
ether=9,0x300,0xd0000,0xd4000,eth0  root=/dev/hda1
```

Nachfolgendes Beispiel ist hingegen falsch:

```
ether = 9, 0x300, 0xd0000, 0xd4000, eth0  root = /dev/hda1
```

Sobald der Kernel erst einmal läuft, kann man jederzeit mit dem Befehl

```
cat /proc/cmdline
```

nachschauen, welche Parameter an den Kernel übergeben wurden.

2.1 LILO (LInux LOader)

LILO (LInux LOader) von Werner Almesberger ist das am häufigsten verwendete Programm zur Übergabe der Parameter. Das Programm hat die Fähigkeit, verschiedene Kernel bzw. Betriebssysteme zu booten. Die meisten Distributionen verwenden standardmäßig LILO, um Linux und z.B. auch Windows zu starten. LILO kann DOS, Windows, OS/2, Linux, FreeBSD, etc. ohne Schwierigkeiten booten und ist zudem äußerst flexibel.

Nach dem Einschalten des Computers wird bei den meisten Installationen LILO gestartet. Drückt der Anwender nach der Meldung `LILO` die TAB-Taste, so gelangt er zum Prompt von LILO. Ansonsten wird nach eine festgelegten Zeit automatisch das Standardsystem gestartet. In der Regel wird hier durch die Eingabe eines `label`-Eintrages aus `/etc/lilo.conf` das zu startende Betriebssystem bzw. Linux Kernel ausgewählt. Typisch sind Labels wie `linux`, `backup` und `msdos`. Falls ein Bootparameter an den Kernel übergeben werden soll, kann man dies an dieser Stelle tun. Der Parameter wird einfach, durch ein Leerzeichen getrennt, an das Label angehängt. Folgendes Beispiel soll dies verdeutlichen:

```
LILO: linux root=/dev/hda1
```

In der Regel möchte man natürlich nicht bei jedem Booten die Parameter per Hand eingeben müssen. Hier hilft die Option `append=`, die der Konfigurationsdatei von LILO hinzugefügt werden kann und deren Parameter dann automatisch an den Kernel übergeben werden. Man muß einfach nur etwas wie

```
append = "foo=bar"
```

in die Datei `/etc/lilo.conf` einfügen. Dieses kann entweder am Anfang der Datei eingefügt werden, so daß die Parameter für alle Abschnitte der Datei gültig sind, oder in den Abschnitt eines bestimmten Kernels, so daß nur diesem die Parameter übergeben werden. Eine ausführliche Beschreibung ist in der ausgezeichneten Dokumentation von LILO zu finden.

2.2 loadlin

Ein anderer weitverbreiteter Linux-Loader ist *loadlin*. Dies ist ein DOS-Programm, das die Fähigkeit besitzt, einen Linux-Kernel vom DOS-Prompt aus zu starten, wobei Bootparameter übergeben werden können.

Sehr nützlich ist die Möglichkeit, Linux von DOS aus zu starten, wenn man bestimmte Hardware besitzt, die von Linux erst dann unterstützt wird, wenn sie von dem der Hardware beiliegenden DOS-Treiber in einen bestimmten Zustand versetzt worden ist. Ein gutes Beispiel sind die Soundblaster-kompatiblen Soundkarten, welche den DOS-Treiber benötigen, um ein paar geheimnisvolle Register ziehen zu können, um die Karte in einen SB-kompatiblen Modus zu bringen. Das Booten von DOS mit dem zur Verfügung stehenden Treiber und das anschließende Laden von Linux mit `LOADLIN.EXE` vom DOS-Prompt aus verhindert das Zurückschalten der Karte in den vorherigen Zustand, was beim erneuten Booten der Fall wäre. So jedoch bleibt die Karte in einem SB-kompatiblen Modus und ist somit unter Linux verwendbar. Auch Plug&Play Hardware kann auf diese Art und Weise initialisiert werden.

Es gibt auch noch andere Programme, die zum Booten von Linux verwendet werden können. Auf dem lokalen Linux-FTP-Server erhält man unter `system/Linux-boot/` die komplette Liste aller verfügbaren Programme.

2.3 Das Hilfsprogramm »rdev«

Es gibt einige Kernel-Bootparameter, deren Standardwerte an bestimmten Stellen im Kernel-Image selbst gespeichert sind. Es gibt ein Hilfsprogramm namens `rdev`, das auf den meisten Systemen installiert ist und das weiß, wo sich diese Werte befinden und wie sie geändert werden können. Dieses Hilfsprogramm kann auch Dinge ändern, die kein Kernel-Bootparameter-Äquivalent besitzen, wie z.B. der standardmäßig verwendete Grafik-Modus.

Das Hilfsprogramm `rdev` ist gewöhnlich auch unter den Namen `swapdev`, `ramsize`, `vidmode` und `rootflags` aufrufbar. Diese Namen zeigen die fünf Änderungsmöglichkeiten durch `rdev` an: das Root-Device, das Swap-Device, die RAM-Disk-Parameter, der Standard-Grafik-Modus sowie die readonly/readwrite-Einstellung vom Root-Device.

Weitere Informationen über `rdev` erhält man nach Eingabe von `rdev -h` oder durch die Lektüre der bereitgestellten Manpage (`man rdev`).

2.4 Sortierung der Parameter durch den Kernel

Die meisten Bootparameter sind folgendermaßen strukturiert:

```
name[=wert_1][,wert_2]...[,wert_11]
```

`name` ist hierbei ein einzigartiges Schlüsselwort, das Aufschluß darüber gibt, für welchen Teil des Kernels die entsprechenden Werte bestimmt sind. Mehrere Bootparameter werden als Liste nach obigem Format an den Kernel übergeben, wobei die einzelnen Parameter durch Leerzeichen getrennt werden. Man beachte das tatsächliche Limit von 11 Parametern. Der bestehende Code kann nur 11 durch Kommas getrennte Parameter pro Schlüsselwort verarbeiten. In ungewöhnlich komplizierten Fällen kann man jedoch dasselbe Schlüsselwort mit 11 zusätzlichen Parametern erneut benutzen, vorausgesetzt, die Setup-Funktion unterstützt dies. Man beachte auch, daß der Kernel die Liste in maximal zehn Ganzzahlen-Parameter und eine anschließende Zeichenfolge unterteilt. Das heißt, man kann nicht wirklich 11 Ganzzahlen bereitstellen; dies ist höchstens durch Konvertierung des 11ten Parameters von einer Zeichenkette in eine Ganzzahl im Treiber selbst möglich.

Die Sortierung findet hauptsächlich in `linux/init/main.c` statt. Zuerst überprüft der Kernel, ob der Parameter zu einem der speziellen Parameter wie `root=`, `ro`, `rw` oder `debug` gehört. Die Bedeutung dieser speziellen Parameter wird im weiteren Verlauf dieser Dokumentation beschrieben.

Der Kernel durchsucht dann eine Liste von Setup-Funktionen, die im `bootsetups`-Array gespeichert ist, um zu sehen, ob ein Treiber oder ein Teil des Kernels für die entsprechende Zeichenkette, wie z.B. `foo`, eine Setup-Funktion `setup_foo()` registriert hat. Würde man dem Kernel die Zeile `foo=3,4,5,6,bar` übergeben, dann würde dieser den `bootsetups`-Array durchgehen, um herauszufinden, ob `foo` registriert ist. Falls dies der Fall wäre, würde der Kernel die Setup-Funktion, die mit `foo` verbunden ist, aufrufen und dieser die Ganzzahlen-Parameter 3, 4, 5 und 6 übergeben, wie sie auf der Kernel-Kommandozeile angegeben wurden. Außerdem würde er ebenfalls die Zeichenkette `bar` übergeben.

2.5 Umgebungsvariablen setzen

Alles, was aussieht wie `foo=bar` und nicht als Setup-Funktion akzeptiert wird, wie oben beschrieben, wird dann als zu setzende Umgebungsvariable interpretiert. Ein Beispiel wäre die Verwendung von `TERM=vt100` oder `BOOT_IMAGE=vmlinuz.bak` als Bootparameter. Diese Umgebungsvariablen werden typischerweise in Initialisierungsskripts abgefragt, um ein große Anzahl von verschiedenen Dingen ein- oder auszuschalten.

2.6 Weitergabe von Parametern zum »init«-Programm

Alle verbleibenden Parameter, die der Kernel nicht selbst verwendet und nicht als Umgebungsvariablen interpretiert, werden zum ersten Prozeß weitergeleitet. Dieser ist normalerweise das `init`-Programm. Meistens wird dem `init`-Programm per Bootparameter mitgeteilt, mit welchem Runlevel Linux gebootet werden soll. So kann `init` angewiesen werden, den Rechner im Ein-Benutzer-Modus zu booten und die üblichen Dämonen nicht zu starten. Um herauszufinden, welche Parameter von der auf Ihrem System installierten Version von `init` akzeptiert werden, lesen Sie bitte die entsprechende Manual Page.

3 Allgemeine, geräteunabhängige Bootparameter

Hierbei handelt es sich um Bootparameter, die mit keinem speziellen Hardware-Treiber verknüpft sind. Statt dessen beeinflussen sie einige bestimmte intere Kernel-Parameter wie z.B. den Umgang mit dem Speicher, mit der RAM-Disk, dem Root-Dateisystem und anderen.

3.1 Root-Dateisystem Optionen

Folgende Optionen beziehen sich alle auf das Vorgehen des Kernels bei der Auswahl und dem Umgang mit dem Root-Dateisystem.

3.1.1 Der Parameter »root=«

Dieser Parameter teilt dem Kernel mit, welches Device beim Booten als Root-Dateisystem benutzt werden soll. Als Standardeinstellung benutzt der Kernel das Device, das auf dem System, auf dem der Kernel erzeugt wurde, das Root-Dateisystem enthielt. Wurde der fragliche Kernel z.B. auf einem System kompiliert, das `/dev/hda1` als Root-Partition verwendete, dann geht der Kernel davon aus, daß sich das Root-Dateisystem auf `/dev/hda1` befindet. Will man diesen Standardwert außer Kraft setzen und z.B. das zweite Diskettenlaufwerk als Root-Device verwenden, würde man `root=/dev/fd1` wählen.

Gültige Root-Devices sind:

- `/dev/hdaN` bis `/dev/hddN`: Partition N auf der ST-506-kompatiblen (IDE) Festplatte `a` bis `d`
- `/dev/sdaN` bis `/dev/sdeN`: Partition N auf der SCSI-kompatiblen Festplatte `a` bis `e`
- `/dev/xdaN` bis `/dev/xdbN`: Partition N auf der XT-kompatiblen Festplatte `a` bis `b`
- `/dev/fdN`: Diskettenlaufwerk mit der Nummer N. N=0 wäre das DOS-Laufwerk »A:« und N=1 wäre »B:«.
- `/dev/nfs`: Dieses ist nicht wirklich ein Device, sondern teilt dem Kernel lediglich mit, daß das Root-Dateisystem über das Netzwerk geholt werden soll.

Die schwierigere und weniger portable numerische Bestimmung der oben genannten, möglichen Devices für Festplatten im Major-/Minor-Format wird auch akzeptiert. So entspricht z.B. Major 8, Minor 3 der Partition `/dev/sda3`, so daß man alternativ `root=0x803` verwenden könnte.

Dies ist einer der wenigen Kernel-Bootparameter, dessen Standard im Kernelimage gespeichert ist, und welcher deshalb mit dem Hilfsprogramm `rdev` geändert werden kann.

3.1.2 Der Parameter »ro«

Wenn der Kernel bootet, wird dabei ein Root-Dateisystem benötigt, um grundlegende Dinge davon abzulesen. Dies ist das Root-Dateisystem, das beim Booten gemountet wird. Wird das Root-Dateisystem jedoch mit Schreibrecht gemountet, kann keine verläßliche Überprüfung der Integrität des Dateisystems vorgenommen werden, weil z.B. gleichzeitig ein anderer Prozeß eine Datei bearbeitet und damit das zu prüfende Dateisystem verändert. Mit der Option `ro` wird der Kernel aufgefordert, das Root-Dateisystem nur mit Leserecht (engl. readonly) zu mounten, so daß Programme zur Konsistenzprüfung des Dateisystems (fsck) mit Sicherheit annehmen können, daß keinerlei halb geschriebene Dateien existieren, während die Überprüfung stattfindet. Kein Programm oder Prozeß kann Dateien des fraglichen Dateisystems verändern, bis es nicht mit Lese-/Schreibrecht wieder gemountet ist.

Auch diese Einstellung ist im Kernelimage gespeichert und kann deshalb mit dem Programm `rdev` verändert werden.

3.1.3 Der Paramater »rw«

Dieser Parameter ist das exakte Gegenteil vom eben Genannten. Hier wird der Kernel nämlich aufgefordert, das Root-Dateisystem mit Lese-/Schreibrecht zu mounten. Die Standardeinstellung ist sowieso das Mounten des Root-Dateisystems mit Lese-/Schreibrecht. Man sollte auf keinen Fall Programme vom Typ »fsck« auf einem Dateisystem laufen lassen, das mit Lese-/Schreibrecht gemountet wurde.

Der bereits oben erwähnte, in der Image-Datei gespeicherte Wert ist der gleiche, der auch für diesen Parameter verwendet wird. Der Zugriff darauf erfolgt über `rdev`.

3.2 Optionen der RAM-Disk-Verwaltung

Folgende Optionen beziehen sich alle auf den Umgang des Kernels mit dem RAM-Disk-Device. Eine RAM-Disk wird hauptsächlich während der Installation einer Linux Distribution verwendet. Außerdem ist die RAM-Disk auch sehr nützlich, wenn der Kernel für den Zugriff auf das Root-Dateisystem zuerst Treiber laden muß, die als Modul kompiliert worden sind.

3.2.1 Das Kommando »ramdisk_start=«

Damit ein Kernel-Image zusammen mit dem komprimierten Image einer RAM-Disk auf einer Diskette gespeichert werden kann, existiert der Parameter `ramdisk_start=<offset>`. Der Kernel kann nicht in das komprimierte Image des RAM-Disk Dateisystems aufgenommen werden, weil der Kernel beginnend mit Block Null auf der Diskette gespeichert werden muß, so daß das BIOS den Bootsektor laden kann. Dann kann der Kernel sich selbst erstmalig booten und damit zum Laufen bringen.

Benutzt man ein unkomprimiertes Image einer RAM-Disk, dann kann der Kernel Teil dieses Images sein, das in die RAM-Disk geladen wird. Die Diskette kann mit LILO gebootet werden. Die beiden können auch getrennt sein wie bei den komprimierten Images.

Verwendet man zwei Disketten, wobei sich auf der ersten der Kernel und auf der zweiten das Image einer RAM-Disk befindet, dann startet die RAM-Disk bei Block Null und es wird ein Offset von Null verwendet. Da dies der Standardwert ist, braucht man das Kommando in Wirklichkeit gar nicht benutzen.

3.2.2 Das Kommando »load_ramdisk=«

Dieser Parameter teilt dem Kernel mit, ob ein Image einer RAM-Disk geladen werden soll oder nicht. Durch die Angabe von `load_ramdisk=1` wird der Kernel veranlaßt, eine Diskette in die RAM-Disk zu laden. Der

Standardwert ist Null, was bedeutet, daß der Kernel nicht versuchen soll, eine RAM-Disk zu laden.

Eine komplette Beschreibung der neuen Bootparameter und deren Verwendung findet man in der Datei `linux/Documentation/ramdisk.txt`. Es wird auch beschrieben, wie dieser Parameter mit `rdev` im Kernel-Image gespeichert werden kann.

3.2.3 Das Kommando »prompt_ramdisk=«

Dieser Parameter teilt dem Kernel mit, ob der Anwender aufgefordert werden soll, die Diskette mit dem Image der RAM-Disk einzulegen. Bei einer Konfiguration mit einer Diskette befindet sich das Image der RAM-Disk auf der gleichen Diskette wie der Kernel, der gerade den Lade-/Bootvorgang beendet hat. Somit wird eine Nachfrage nicht benötigt. In diesem Fall kann man `prompt_ramdisk=0` verwenden. Bei einer Konfiguration mit zwei Disketten wird man die Möglichkeit des Diskettentauschs benötigen. Somit kann `prompt_ramdisk=1` verwendet werden. Da dies der Standardwert ist, muß er eigentlich nicht angegeben werden. Früher haben raffinierte Anwender die Option `vga=ask` von LILO verwendet, um den Bootprozeß zeitweilig zu stoppen und ein Wechsel von der Boot- zur Rootdiskette zu ermöglichen.

Eine komplette Beschreibung der neuen Bootparameter und deren Benutzung findet man in der Datei `linux/Documentation/ramdisk.txt` Es wird auch beschrieben, wie dieser Parameter bestimmt und via `rdev` im Kernel-Image gespeichert werden kann.

3.2.4 Das Kommando »ramdisk_size=«

Zwar stimmt es, daß die RAM-Disk je nach Bedarf dynamisch wächst, jedoch gibt es eine Obergrenze für ihre Größe, so daß nicht der gesamte Speicher verbraucht wird und es zu Problemen kommt. Der Standardwert ist 4096 (4 MB), was den meisten Ansprüchen genügen sollte. Mit diesem Bootparameter kann man den Standardwert verringern oder vergrößern.

Eine komplette Beschreibung der neuen Bootparameter und deren Benutzung findet man in der Datei `linux/Documentation/ramdisk.txt` Es wird auch beschrieben, wie dieser Parameter bestimmt und via `rdev` im Kernel-Image gespeichert werden kann.

3.2.5 Das Kommando »ramdisk=« (veraltet)

Dieser Parameter ist veraltet und sollte nicht verwendet werden, es sei denn für die Kernelversion v1.3.47 oder ältere. Die Parameter, die heute für das RAM-Disk-Device verwendet werden sollten, sind oben beschrieben.

Mit dem Parameter wird die Größe RAM-Disk-Devices in KByte angeben. Würde man z.B. ein Root-Dateisystem auf einer 1,44 MB Diskette in ein RAM-Disk-Device laden wollen, würde man folgendes Kommando verwenden:

```
ramdisk=1440
```

Dies ist einer der wenigen Kernel-Bootparameter, dessen Standardwert im Kernel-Image gespeichert ist und der somit mit dem Hilfsprogramm `rdev` verändert werden kann.

3.2.6 Das Kommando »noinitrd«

Kernel v2.x und neuere verfügen über ein Feature, bei dem das Root-Dateisystem anfangs eine RAM-Disk sein kann und der Kernel auf diesem RAM-Image `/linuxrc` ausführt. Dieses Feature wird typischerweise dazu verwendet, das Laden von Modulen zu ermöglichen, die zum Mounten des eigentlichen Root-Dateisystems benötigt werden. So können z.B. die Module des SCSI-Treibers von dem Image der RAM-Disk geladen werden und anschließend wird das eigentliche Root-Dateisystem auf einer SCSI-Festplatte gemountet.

Der eigentliche `noinitrd`-Parameter bestimmt, was mit den initrd-Daten passiert, nachdem der Kernel gebootet wurde. Wenn dieser Parameter gesetzt wird, werden die Daten nicht auf eine RAM-Disk konvertiert, sondern es kann auf diese Daten unter `/dev/initrd` zugegriffen werden. Dieser Zugriff ist nur einmal möglich, bevor der Speicher an das System zurückgegeben wird. Umfassende Informationen über die Benutzung der initial RAM-Disk erhält man unter `linux/Documentation/initrd.txt`. Zusätzlich sollten die neuesten Versionen von LILO und loadlin weitere nützliche Informationen enthalten.

3.3 Bootparameter für den Umgang mit dem Speicher

Folgende Parameter ändern sich je nachdem, wie Linux den physikalischen oder virtuellen Systemspeicher erkennt oder mit ihm umgeht.

3.3.1 Der Parameter »mem=«

Dieser Parameter dient zwei verschiedenen Zwecken: Der ursprüngliche Zweck war, die Größe des installierten Speichers oder einen kleineren Wert, falls man die Größe des für Linux verfügbaren Speichers verringern wollte, zu spezifizieren. Der andere, kaum verwendete Zweck ist die Bestimmung von `mem=nopentium`, was den Linux-Kernel auffordert, das 4 MB Seitentabellen-Performance-Feature nicht zu verwenden.

Während des Bootvorganges ermittelt Linux durch einen BIOS-Aufruf die Größe des installierten Speichers. Leider ist die Größe, die dieser Aufruf zurückliefern kann, auf 64 MB begrenzt. Erinnert uns das irgendwie an das 1024 Zylinder Limit von IDE-Festplatten? Sind mehr als 64 MB RAM installiert, kann man Linux mit diesem Bootparameter mitteilen, wieviel Speicherplatz vorhanden ist. Hier ein Zitat von Linus über die Verwendung des `mem=` Parameters:

> Der Kernel wird alle `mem=xx` Parameter akzeptieren, die man ihm übergibt. Falls sich allerdings herausstellt, daß man gelogen hat, wird der Kernel früher oder später schrecklich abstürzen. Der Parameter legt die höchste ansprechbare RAM-Adresse fest, so daß z.B. `mem=0x1000000` bedeutet, daß man 16 MB Speicher besitzt. Für einen Rechner mit 96 MB wäre der Bootparameter `mem=0x6000000`. Teilt man Linux mit, daß es über mehr als den tatsächlich vorhandenen Speicherplatz verfügt, dann wird dies schlimme Folgen haben; vielleicht nicht sofort, aber irgendwann mit Sicherheit.

Man beachte, daß der übergebene Wert keine Hexadezimalzahl sein muß und daß die Endungen `k` bzw. `M` zur Bestimmung von Kilobytes bzw. Megabytes verwendet werden können, wobei Groß- oder Kleinschreibung keine Rolle spielen. `k` verursacht eine Verschiebung des Wertes um 10 Bit, während `M` eine Verschiebung um 20 Bit bewirkt. Für einen Rechner mit z.B. 128 MB RAM würde man `mem=128m` verwenden.

3.3.2 Der Parameter »swap=«

Hier wird dem Anwender die Feineinstellung eines Teils der virtuellen Speicherparameter ermöglicht, die mit Diskswapping zu tun haben. Folgende acht Parameter werden akzeptiert:

```
MAX_PAGE_AGE
PAGE_ADVANCE
PAGE_DECLINE
PAGE_INITIAL_AGE
AGE_CLUSTER_FRACT
AGE_CLUSTER_MIN
PAGEOUT_WEIGHT
BUFFEROUT_WEIGHT
```

Interessierten Hackern sei die Lektüre von `linux/mm/swap.c` empfohlen. Auch sollten sie einen Blick in `/proc/sys/vm` werfen. Das Kernel enthält nützliche Dokumentationen zu diesem Thema im Verzeichnis `linux/Documentation/vm/`.

3.3.3 Der Parameter »buff=«

Ähnlich dem `swap=`-Parameter wird dem Anwender die Feineinstellung einiger der Parameter für die Buffer-Speicherverwaltung ermöglicht. Folgende sechs Parameter werden akzeptiert:

```
MAX_BUFF_AGE
BUFF_ADVANCE
BUFF_DECLINE
BUFF_INITIAL_AGE
BUFFEROUT_WEIGHT
BUFFERMEM_GRACE
```

Interessierten Hackern sei die Lektüre von `linux/mm/swap.c` empfohlen. Auch sollten sie einen Blick in `/proc/sys/vm` werfen. Das Kernel enthält nützliche Dokumentationen zu diesem Thema im Verzeichnis `linux/Documentation/vm/`.

3.4 Bootparameter für das NFS-Root-Dateisystem

Linux unterstützt Systeme wie laufwerkslose Workstations, die ihr Root-Dateisystem per NFS (Network FileSystem) beziehen. Diese Parameter werden dazu verwendet, der laufwerkslosen Workstation mitzuteilen, von welchem Rechner sie ihr System erhält. Man beachte, daß der Parameter `root=/dev/nfs` benötigt wird. Detaillierte Informationen über die Verwendung eines NFS-Root-Dateisystems findet man in der Datei `linux/Documentation/nfsroot.txt`. Es wird empfohlen, diese Datei zu lesen, da folgende Informationen lediglich aus einer Zusammenfassung dieser Datei bestehen.

3.4.1 Der Parameter »nfsroot=«

Dieser Parameter teilt dem Kernel mit, welcher Rechner, welches Verzeichnis und welche NFS-Optionen für das Root-Dateisystem verwendet werden sollen. Der Parameter ist folgendermaßen aufgebaut:

```
nfsroot=[<server-ip>:]<root-verz>[,<nfs-optionen>]
```

Wird der `nfsroot`-Parameter nicht auf der Kommandozeile angegeben, dann wird standardmäßig `/tftpboot/%s` verwendet. Andere mögliche Optionen sind:

<server-ip>

Bestimmt die IP-Adresse des NFS-Servers. Fehlt dieses Feld, dann wird die von der `nfsaddrs`-Variablen (siehe unten) bestimmte Default-Adresse verwendet. Dieser Parameter wird z.B. dazu verwendet, die Benutzung mehrerer Server für RARP und NFS zu ermöglichen. Normalerweise kann dieses Feld leer bleiben.

<root-verz>

Name des Verzeichnisses auf dem Server, das als root gemountet werden muß. Ist in der Zeichenkette ein `%s`-Token enthalten, dann wird der Token durch die ASCII-Darstellung der IP-Adresse des Clients ersetzt.

<nfs-optionen>

Standard-NFS-Optionen. Alle Optionen sind durch Kommas getrennt. Fehlt das Optionen-Feld, werden folgende Standardwerte verwendet:

```
port            = wie vom Portmap-Daemon angegeben
rsize           = 1024
wsize           = 1024
timeo           = 7
retrans         = 3
acregmin        = 3
acregmax        = 60
acdirmin        = 30
acdirmax        = 60
flags           = hard, nointr, noposix, cto, ac
```

3.4.2 Der Parameter »nfsaddrs=«

Dieser Bootparameter bestimmt die verschiedenen Adressen der Netzwerkschnittstellen, die zur Kommunikation über's Netzwerk benötigt werden. Wird dieser Parameter nicht angegeben, dann versucht der Kernel unter Verwendung von RARP und/oder BOOTP, diese Parameter herauszufinden. Der Syntax sieht folgendermaßen aus:

```
nfsaddrs=<meine-ip>:<serv-ip>:<gw-ip>:<netmask>:<name>:<dev>:<auto>
```

<meine-ip>

IP-Adresse des Clients. Ist dieses Feld leer, wird die Adresse entweder durch RARP oder durch BOOTP bestimmt. Welches Protokoll verwendet wird, hängt von dem `<auto>` Parameter ab und davon, was während der Kernelkonfiguration aktiviert wurde. Ist dieser Parameter nicht leer, dann wird weder RARP noch BOOTP benutzt.

<serv-ip>

IP-Adresse des NFS-Servers. Wird RARP zur Bestimmung der Client-Adresse verwendet und ist dieser Parameter *nicht* leer, dann werden nur Antworten vom festgelegten Server akzeptiert. Zur Verwendung verschiedener RARP- und NFS-Server bestimmt man hier den RARP-Server oder läßt das Feld leer und legt den NFS-Server mit dem `nfsroot`-Parameter fest (siehe oben). Bleibt dieser Eintrag aus, dann wird die Adresse des Servers verwendet, welcher auf die RARP- oder BOOTP-Anfrage geantwortet hat.

<gw-ip>

IP-Adresse eines Gateways, falls der Server sich in einem anderen Subnetz befindet. Ist dieser Eintrag leer, dann wird kein Gateway verwendet und es wird angenommen, daß sich der Server im lokalen Netzwerk befindet, außer wenn durch BOOTP ein Wert empfangen wird.

<netmask>

Netmask für die lokale Netzwerkschnittstelle. Ist dieser Eintrag leer, dann wird die Netmask von der Client-IP-Adresse abgeleitet, wenn nicht durch BOOTP ein Wert empfangen wird.

<name>

Name des Clients. Bleibt dieses Feld leer, dann wird die Client-IP-Adresse in ASCII-Notation verwendet oder der durch BOOTP empfangene Wert.

<dev>

Name des zu verwendenden Netzwerk-Devices. Ist dieses Feld leer, dann werden alle Devices für RARP-Anfragen verwendet und das erste Device für BOOTP. Für NFS wird das Device benutzt, von dem entweder RARP- oder BOOTP-Antworten empfangen wurden. Besitzt man nur ein Device, kann man dieses Feld getrost leer lassen.

<auto>

Zu verwendende Methode für die automatische Konfiguration. Ist dieses entweder `rarp` oder `bootp`, dann wird das angegebene Protokoll verwendet. Ist der Wert `both` oder leer, dann werden beide Protokolle in dem Umfang verwendet, wie es ihnen während der Kernelkonfiguration ermöglicht wurde. Der Eintrag `none` schließt die automatische Konfiguration aus. In diesem Fall müssen alle notwendigen Werte in den vorherigen Feldern bestimmt werden.

Der Parameter `<auto>` kann alleine als Wert für den Parameter `nfsaddrs` erscheinen, wobei die ganzen »:«-Zeichen davor entfallen. In diesem Fall wird die automatische Konfiguration verwendet. Jedoch ist der Wert `none` in diesem Fall nicht verfügbar.

3.5 Weitere verschiedene Kernel-Bootparameter

Diese verschiedenen Bootparameter erlauben dem Benutzer die Feineinstellung bestimmter interner Parameter.

3.5.1 Der Parameter »debug«

Mittels der Funktion `printk()` schickt der Kernel wichtige und weniger wichtige Nachrichten an den Administrator. Wird die Nachricht als wichtig angesehen, dann wird die Funktion `printk()` eine Kopie auf der aktuellen Konsole ausgeben und sie an `klogd()` übergeben, so daß sie auf Festplatte gespeichert wird. Der Grund für die Ausgabe wichtiger Nachrichten auf der Konsole und gleichzeitiger Protokollierung auf der Festplatte liegt darin, daß unter unglücklichen Umständen wie z.B. einem Plattenausfall die Nachricht nicht zur Festplatte gelangt und somit verloren geht.

Der Grenzwert dafür, was als wichtig oder nicht wichtig betrachtet wird, wird von der Variablen `console_loglevel` festgelegt. Standardmäßig wird alles, was wichtiger ist als `DEBUG` (Level 7), auf der Konsole ausgegeben. Die verschiedenen Level sind in der Include-Datei `kernel.h` definiert. Das Festlegen von `debug` als Bootparameter setzt den Grenzwert der Konsole auf 10, so daß *alle* Mitteilungen des Kernels auf der Konsole erscheinen.

Der Grenzwert der Konsole kann normalerweise auch bei der Ausführung mittels einer Option des Programmes `klogd` festgelegt werden. Informationen darüber kann man der Manpage zu der auf dem System installierten Version entnehmen.

3.5.2 Der Parameter »init=«

Der Kernel wird beim Booten immer das `init`-Programm starten, welches getty-Programme startet, »rc«-Skripts laufen läßt, u.ä. und somit den Rechner für die Benutzer einrichtet. Der Kernel schaut zuerst nach `/sbin/init`, dann nach `/etc/init` und als letzte Möglichkeit wird er versuchen, `/bin/sh` zu verwenden (möglicherweise auf `/etc/rc`). Wurde z.B. das `init`-Programm verfälscht und somit das Booten unmöglich gemacht, dann kann man einfach am Bootprompt `init=/bin/sh` verwenden, was beim Booten direkt eine Shell öffnet und somit ein Ersetzen des verfälschten Programms ermöglicht.

3.5.3 Der Parameter »kbd-reset«

Normalerweise wird bei i386 basierten Rechnern der Tastaturkontroller vom Linux Kernel beim Booten nicht initialisiert, da diese Aufgabe vom BIOS übernommen wird. Aber, wie es nicht anders zu erwarten war, machen dieses nicht alle Rechner standardmäßig. Die Verwendung dieses Parameters kann helfen, wenn es Probleme mit der Tastatur gibt. Der Parameter führt einfach dazu, daß beim Booten der Kontroller initialisiert wird.

3.5.4 Der Parameter »maxcpus=«

Die mit dem Parameter übergebene Zahl limitiert die maximale Anzahl von CPUs, die im SMP-Modus aktiviert werden. Die Verwendung der Zahl »0« hat die gleiche Funktion wie der `nosmp` Parameter.

3.5.5 Der Parameter »mca-pentium«

Die IBM Model 95 Microchannel Rechner scheinen sich beim dem Test aufzuhängen, den Linux durchführt, um den Typ der Kopplung des mathematischen Koprozessors mit der CPU zu ermitteln. Da alle Pentium CPUs einen eingebauten Koprozessor besitzen, kann dieser Test und damit das beschriebene Probleme durch die Verwendung dieses Parameter unterdrückt werden.

3.5.6 Der Parameter »md=«

Wenn sich das Root-Dateisystem sich auf einem *Multiple Device* befindet, kann dieser Parameter verwendet werden, um dem Kernel das Layout des Multiple Devices mitzuteilen. Hierfür muß natürlich die Bootunterstützung einkompiliert sein. Das Format dieses Parameter sieht so aus:

```
md=md_device_num,raid_level,chunk_size_factor,fault_level,dev0,dev1,...,devN
```

Siehe auch die Datei `linux/Documentation/md.txt`. Hierbei ist `md_device_num` die Nummer des md Devices; 0 würde für `md0`, 1 für `md1` usw. stehen. Für `raid_level` kann -1 (Linearer Modus) oder 0 (Striped Modus) verwendet werden. Andere Modi werden zur Zeit nicht unterstützt. Die Option `chunk_size_factor` kann nur für RAID-0 und RAID-1 benutzt werden. Mit ihr wird die Chunk Size gesetzt; aus dem übergebenen Wert wird die Chunk Size berechnet, in dem die PAGE_SIZE um den angegebenen Wert nach links geshiftet wird. Mit `fault_level` wird die maximale Anzahl von Fehlern bei RAID-1 festgelegt; da von RAID-1 zur Zeit nicht gebootet werden kann, hat diese Option keine Funktion. `dev0,dev1,...,devN` ist eine durch Kommata getrennte Liste von Devices, aus denen das md Device gebildet wird, z.B. `/dev/hda1,/dev/hdc1,/dev/sda1`.

3.5.7 Der Parameter »no387«

Einige i387 Koprozessoren haben Fehler, die sich bei der Verwendung im 32 Bit Protected Mode zeigen. Einige der frühen i387 Koprozessoren von ULSI verursachen z.B. massive Totalsperren während der Ausführung von Fließkomma-Berechnungen, was offensichtlich ein Bug in den FRSAV/FRRESTOR Anweisungen ist. Die Verwendung des Bootparameters `no387` veranlaßt Linux, den mathematischen Koprozessor zu ignorieren, sogar wenn einer vorhanden ist. Natürlich muß der Kernel dann so kompiliert werden, daß sich die Emulation eines Koprozessors im Kernel befindet. Dies ist möglicherweise auch dann sinnvoll, wenn man einen dieser *wirklich* alten i386er hat, die einen 80287 FPU verwenden können, da Linux keine 80287 FPUs unterstüzt.

3.5.8 Der Parameter »no-hlt«

Die Familie der i386 CPUs und deren Nachfolger verfügen über die Anweisung »hlt«, die der CPU mitteilt, daß nichts geschehen wird, solange nicht ein externes Gerät (Tastatur, Modem, Platte, etc.) die CPU auffordert, eine Aufgabe auszuführen. Dieses erlaubt der CPU in einen Modus zu schalten, in dem weniger Energie verbraucht wird. In diesem Modus verharrt sie wie ein Zombie, bis sie von einem externen Gerät, gewöhnlich durch einen Interrupt, geweckt wird. Einige der frühen i486DX-100 CPUs hatten insofern ein Problem mit der Anweisung »hlt«, daß sie nach deren Ausführung nicht mehr verläßlich in den Betriebsmodus zurückkehren konnten. Durch die Benutzung der Anweisung `no-hlt` wird Linux mitgeteilt, einfach eine Endlosschleife zu durchlaufen, wenn es nichts anderes zu tun gibt und die CPU *nicht* zu stoppen, wenn es keine aktiven Prozesse gibt. Dieses ermöglicht Benutzern mit solchen »defekten« CPUs die Verwendung von Linux, obwohl sie gut beraten wären, sich durch eine möglicherweise noch vorhandene Garantie einen Ersatz zu suchen.

3.5.9 Der Parameter »no-scroll«

Die Angabe dieses Parameters beim Booten setzt Bildlauf-Features außer Kraft, die die Verwendung von Braille-Terminals erschweren.

3.5.10 Der Parameter »noapic«

Durch die Verwendung dieses Parameters wird einem SMP-Kernel mitgeteilt, nicht die erweiterten Funktionen des Interrupt Kontrollers eines Mehrprozessorrechners zu benutzen. Weitere Informationen hierzu sind in der Datei `linux/Documentation/IO-APIC.txt` zu finden.

3.5.11 Der Parameter »nosmp«

Mittels dieses Parameters kann ein SMP-Kernel angewiesen werden, auf einem SMP-Rechner nur einen Prozessor zu verwenden. Typischerweise wird dieser Parameter nur fürs Debugging benutzt oder um herauszufinden, ob ein bestimmtes Problem durch SMP verursacht wird.

3.5.12 Der Parameter »panic=«

Für den unwahrscheinlichen Fall einer Kernel-Panik wird für gewöhnlich gewartet, bis jemand vorbeikommt, die Nachricht der Panik auf dem Bildschirm entdeckt und den Rechner neu bootet. Bei einer Kernel-Panik handelt es sich um einen internen Fehler, der von Kernel erkannt und als ernst genug erachtet wurde, um sich laut zu beschweren und dann alles zu stoppen. Falls sich ein Rechner jedoch unbewacht in einer abgelegenen Ecke befindet, mag es wünschenswert sein, daß automatisch ein Reset stattfindet, so daß der Rechner wieder zum normalen Betrieb zurückkehrt. Die Angabe von `panic=30` beim Booten hätte z.B. zur Folge, daß der Kernel 30 Sekunden nach der Kernel-Panik versuchen würde, sich selbst zu booten. Der Standardwert ist Null und führt zum Standardverhalten, das darin besteht, endlos zu warten.

Man beachte, daß dieser Zeitlimit-Wert auch durch die `/proc/sys/kernel/panic` sysctl-Schnittstelle gelesen und gesetzt werden kann.

3.5.13 Der Parameter »pirq=«

Diese Parameter dient dazu, einem SMP-Kernel Informationen über die Interrupts der PCI-Steckplätze für unbekannte oder sich auf der schwarzen Liste befindliche SMP Motherboards zu übergeben. Weitere Informationen hierzu sind in der Datei `linux/Documentation/IO-APIC.txt` zu finden.

3.5.14 Der Parameter »profile=«

Kernel-Entwickler können eine Option aktivieren, die es ihnen erlaubt, herauszufinden, wie und wo der Kernel seine CPU-Zyklen einsetzt, um das äußerste an Effizienz und Leistung herauszuholen. Mit dieser Option kann man die Profil-Verschiebungszählung beim Booten bestimmen. Normalerweise wird diese auf 2 gesetzt. Man kann den Kernel auch mit automatisch aktivierter Profiling kompilieren. In jedem Fall braucht man ein Tool wie `readprofile.c`, das die Ausgabe von `/proc/profile` verwenden kann.

3.5.15 Der Parameter »reboot=«

Diese Option kontrolliert die Art des Reboots, die Linux beim Reset des Rechners ausführt. Seit Version 2.0.x ist der Standard ein »kalter« Neustart (kompletter Reset, BIOS macht einen Speicher-Check, etc.) statt eines »warmen« Neustarts (kein kompletter Reset, kein Speicher-Check). Die Voreinstellung wurde in einen Kaltstart geändert, da dieser bei billiger/kaputter Hardware, die es nicht schafft, neu zu booten, wenn ein Warmstart erforderlich wäre, normalerweise funktioniert. Zum Wiederherstellen des alten Zustands, nämlich der Verwendung eines Warmstarts, verwendet man `reboot=w`. Tatsächlich funktioniert auch jedes beliebige Wort, das mit `w` beginnt.

Warum sollte man sich darum kümmern? Einige Platten-Kontroller mit eingebautem Cache-Speicher können einen Warmstart erkennen und Daten aus dem Cache auf die Festplatte schreiben. Nach einem Kaltstart würde der Kontroller eventuell zurückgesetzt werden und die noch auf die Festplatte zu schreibenden Daten im Cache würden verloren gehen. Andere Anwender haben Systeme, die recht lange für den Speicher-Check brauchen oder die ein SCSI BIOS enthalten, das nach einem Kaltstart länger für die Initialisierung braucht.

3.5.16 Der Parameter »reserve=«

Dieser wird dazu benutzt, Teile der I/O Ports vor der Überprüfung zu *schützen*. Das Kommando ist folgendermaßen aufgebaut:

```
reserve=iobase,extent[,iobase,extent]...
```

Bei einigen Rechnern mag es notwendig sein, die automatische Hardwareerkennung der Gerätetreiber davon abzuhalten, in einer bestimmten Region nach Geräten zu suchen. Der Grund dafür kann z.B. schlecht entwickelte Hardware sein, die den Bootvorgang *stoppt* (wie z.B. einige Ethernetkarten), irrtümlicherweise erkannte Hardware, Hardware, deren Zustand durch eine frühere Erkennung geändert wurde oder einfach Hardware, die vom Kernel nicht initialisiert werden soll.

Der Bootparameter `reserve` geht dieses Problem dadurch an, daß er einen I/O Port-Bereich angibt, der nicht geprüft werden soll. Diese Region wird in der Registrationstabelle des Kernels für Ports so behandelt, als ob unter der Adresse bereits ein Gerät mit dem Namen `reserved` gefunden wurde. Man beachte, daß dieser Mechanismus für die meisten Rechner nicht notwendig sein dürfte. Er ist nur bei Problemen und in besonderen Fällen erforderlich.

Die I/O Ports in dem angegebenen Bereich sind gegen eine Geräteerkennung geschützt, bei der zuerst die Funktion `check_region()` ausgeführt wird, statt blind einen I/O Bereich zu prüfen. Dieses wird dann eingesetzt, wenn Treiber bei der Erkennung z.B. durch eine NE2000-Karte hängenbleiben oder irrtümlich andere Geräte als eigene erkannt werden. Ein korrekter Gerätetreiber sollte keine reservierten Bereiche prüfen, wenn nicht ein anderer Bootparameter dieses ausdrücklich verlangt. Das bedeutet, daß `reserve` meistens zusammmen mit einem anderen Bootparameter verwendet wird. Wenn man also einen reservierten Bereich festlegt, der ein bestimmtes Gerät schützen soll, dann muß man im allgemeinen explizit eine Erkennung dieses Gerätes erzwingen. Die meisten Treiber ignorieren die Registrierungstabelle für Ports, wenn ihnen eine bestimmte Adresse genannt wird.

Die Bootzeile

```
reserve=0x300,32  blah=0x300
```

hält z.B. alle Gerätetreiber, mit Ausnahme des Treibers für »blah« davon ab, 0x300-0x31f zu prüfen.

Wie üblich bei Boot-Argumenten gibt es ein Limit von elf Parametern, d.h. man kann nur fünf reservierte Bereiche pro `reserve` Schlüsselwort bestimmen. Bei ungewöhnlich komplizierten Anfragen sind jedoch mehrere `reserve` Argumente möglich.

3.5.17 Der Parameter »vga=«

Man beachte, daß es sich hierbei nicht um einen wirklichen Bootparameter handelt. Es ist vielmehr eine Option, die von LILO interpretiert wird und nicht vom Kernel wie all die anderen Bootparameter. Sie wird jedoch so häufig verwendet, daß sie an dieser Stelle eine Erwähnung verdient. Sie kann auch durch die Verwendung von `rdev -v` festgelegt werden und ebenso durch `vidmode` auf der Datei `vmlinuz`. Dieses ermöglicht dem Setup-Code die Benutzung des Video-BIOS zur Änderung des Standard-Anzeige-Modus vor dem tatsächlichen Booten des Linux-Kernels. Typische Modi sind 80x50, 132x44 usw. Man verwendet diese Option am besten, indem man mit `vga=ask` beginnt, worauf man eine Liste verschiedener Modi erhält, die man mit dem Grafik-Adapter benutzen kann, bevor man den Kernel bootet. Hat man einmal eine Nummer aus obiger Liste gewählt, kann man sie später anstelle von `ask` setzen. Weitere Informationen findet man in der Datei `linux/Documentation/svga.txt`, die mit allen neueren Kernel-Versionen ausgeliefert wird.

Man beachte, daß neuere Kernelversionen (v2.1 und höher) den Setup-Code, der den Grafik-Modus ändert, als Option enthalten. Diese ist aufgelistet als *Video mode selection support*, d.h. man muß diese Option aktivieren, um dieses Feature verwenden zu können.

4 Bootparameter für die Kontrolle des PCI-Bus Verhaltens

Der Parameter »pci=«, der in v2.0 Kernel noch nicht verfügbar war, kann benutzt werden, um das Verhalten beim Testen der PCI Devices und der PCI Devices selbst zu verändern. In der Sourcedatei `linux/drivers/pci/pci.c` werden zuerst die von der Computerarchitektur unabhängigen »pci=«-Parameter ausgewertet. Die übrig gebliebenen Parameter werden dann von `linux/arch/xxx/kernel/bios32.c` behandelt, wobei `xxx` für die Architektur des Rechners steht. Die folgenden Optionen beziehen sich auf die i386-Architektur.

4.1 Die Parameter »pci=bios« und »pci=nobios«

Standardmäßig wird der PCI-Bus und seine Devices vom BIOS des Motherboars getestet und konfiguriert. Mit diesem Parameter kann festgelegt werden, ob das BIOS diese Aufgabe übernehmen soll oder nicht.

4.2 Die Parameter »pci=conf1« und »pci=conf2«

Wenn der PCI Direct Modus eingeschaltet ist, kann durch die Verwendung dieser Parameter entweder die Konfiguration Typ 1 oder Typ 2 eingeschaltet werden. Hierdurch wird implizit die Option »pci=nobios« aktiviert.

4.3 Der Parameter »pci=io=«

Falls Sie eine Meldung wie »PCI: Unassigned IO space for ...« erhalten sollten, müssen Sie eventuell einen I/O-Wert mit diesem Parameter setzen. Im Source des Linux Kernels ist zu diesem Parameter zu lesen:

> Mehrere BIOSe vergessen, I/O-Bereichen Adressen zuzuweisen. Wir versuchen, dieses hiermit zu beheben, wobei wir davon ausgehen, das beginnend mit 0x5800 freie Adressen existieren. Dies ist keine gut Lösung; aber bis wir ein besseres Ressourcenmanagement integriert haben, ist dieses die einzige einfache Lösung.

4.4 Der Parameter »pci=nopeer«

Hiermit wird die standardmäßige Peer Bridge Fehlerbehebung abgeschaltet, die gemäß dem Kernel Source folgendes macht:

> Für den Fall, daß Peer Host Bridges existieren, scanne den Bus hinter ihnen. Obwohl verschiedene Quellen behaupten, daß eine Host Bridge den Header Typ 1 haben und eine Bus Nummer wie für eine PCI2PCI Bridge zugewiesen bekommen haben sollten, stimmt dieses in der Realität nicht. Die Bus Nummer wird vom BIOS meistens auf den ersten freien Wert gesetzt.

4.5 Der Parameter »pci=nosort«

Durch die Verwendung dieses Parameters wird der Kernel aufgefordert, die PCI Devices während der Testphase nicht zu sortieren.

4.6 Der Parameter »pci=off«

Sämtliche PCI-Tests werden mit diesem Parameter abgeschaltet. Jeder Gerätetreiber, der Gebrauch von den PCI-Funktionen macht, um die Hardware zu finden und zu initialisieren, wird mit großer Wahrscheinlichkeit nicht mehr funktionieren.

4.7 Der Parameter »pci=reverse«

Hiermit wird die Reihenfolge der PCI Devices umgekehrt.

5 Bootparameter für Video Frame Buffer Treiber

Der Parameter »video=«, der in v2.0.x Kerneln noch nicht verfügbar war, kann benutzt werden, wenn im Kernel die Frame Buffer Device Abstraktionsschicht einkompiliert worden ist. Wenn dieses auch kompliziert zu sein scheint, es ist garnicht so schlimm.

Im Prinzip bedeutet dieses, daß Programme wie ein X11-Server, die den Grafikmodus einer Grafikkarte nutzen, nicht für jede Grafikkarte speziell angepaßt werden müssen. Statt dessen enthält der Kernel für jede Grafikkarte einen passenden Treiber und bietet den Anwendungen eine einheitliche Programmierschnittstelle. Man benötigt dann z.B. nicht mehr für jede Grafikkarte einen eigenen X11-Server (`XF86_S3`, `XF86_SVGA` usw.) sondern nur noch einen einzigen für die Schnittstelle des Kernels (`XF_FBDev`).

Vergleichbar ist dieses Vorgehen z.B. mit den Netzwerkkartentreibern. Auch hier enthält der Kernel für alle unterstützten Netzwerkkarten einen Treiber und bietet den Anwendungen eine einheitliche Programmierschnittstelle, so daß ein Programm automatisch mit allen unterstützten Netzwerkkarten funktioniert. Welche Netzwerkkarte in einem speziellen System Verwendung findet, ist für die Anwendung unwichtig.

Typischerweise ist das Format dieses Parameters:

```
video=name:option1,option2,...
```

Hierbei ist `name` der Name einer allgemeinen Option oder eines Frame Buffer Treibers. Der »video=« Parameter wird zur weiteren Verarbeitung von `linux/init/main.c` an `linux/drivers/video/fbmem.c` weitergegeben. Hier wird der Parameter zuerst auf einige allgemeine Optionen untersucht, bevor versucht wird, ein Treiber mit diesem Namen zu finden. Ist erst einmal ein passender Name eines Treiber gefunden worden, wird die durch Kommata getrennte Liste von Optionen an diesen Treiber zur weiteren Verarbeitung übergeben. Eine Liste von gültigen Namen von Treiber kann in dem Array `fb_drivers` in der oben erwähnten Datei `fbmem.c` gefunden werden.

Informationen über die Optionen, die die einzelnen Treiber unterstützen, können in dem Verzeichnis `linux/Documentation/fb/` gefunden werden. Zur Zeit (v2.2) werden dort nur einige wenige beschrieben. Unglücklicherweise ist die Anzahl von Grafiktreibern und die Anzahl von Optionen, die diese unterstützen, so groß, daß sie hier nicht aufgelistet werden können.

Falls es für Ihre Grafikkarte keine Dokumentation gibt, müssen Sie die Informationen der verfügbaren Optionen direkt aus dem entsprechenden Treiber gewinnen. Wechseln Sie dazu in das Verzeichnis `linux/drivers/video/` und schauen Sie sich die passende `xxxfb.c` Datei an, wobei Sie `xxx` durch den Namen Ihrer Grafikkarte ersetzen müssen. In dieser Datei sollten Sie dann nach einer Funktion suchen, deren Name `_setup` enthält. In dieser Funktion sollten die Optionen aufgelistet sein, die der Treiber unterstützt.

5.1 Der Parameter »video=map:...«

Mit diesem Parameter kann das Konsole - Frame Buffer Device Mapping gesetzt bzw. geändert werden. Eine durch Kommata getrennte Liste von Zahlen setzt das Mapping. Der Wert von Option (n) wird als Frame Buffer Device Nummer für Konsole (n) verwendet.

5.2 Der Parameter »video=scrollback:...«

Eine Zahl, die hinter dem Doppelpunkt angegeben werden muß, legt die Größe des Speichers fest, der für den Scrollback Buffer reserviert wird. Durch diesen Scrollback Buffer kann der Anwender mittels Shift und Page Up oder Page Down den Bildschirminhalt »zurückblättern«. Durch das Anhängen des Buchstabens »k« oder »K« ans Ende der Zahl wird dem Treiber mitgeteilt, daß die übergebene Zahl die Größe in Kilobytes und nicht in Bytes angibt.

5.3 Der Parameter »video=vc:...«

Eine Zahl bzw. ein Zahlenbereich (z.B. `video=vc:2-5`) spezifiziert die erste bzw. die ersten und die letzte Frame Buffer Virtual Console. Die Verwendung dieses Parameters hat außerdem den Effekt, daß die Frame Buffer Konsole nicht die Standardkonsole ist.

6 Bootparameter für SCSI-Peripheriegeräte

Dieser Abschnitt enthält eine Beschreibung der Bootparameter, die zur Weitergabe von Informationen über die installierten SCSI-Host-Adapter und SCSI-Geräte verwendet werden.

6.1 Parameter für Mid-Level-Treiber

Das SCSI-System von Linux gliedert sich in zwei Teile. Für jeden SCSI-Adapter gibt es einen speziellen Low-Level-Treiber, der dessen Funktionalität über eine definierte Schnittstelle den Mid-Level-Treibern zugänglich macht. Für jeden SCSI-Gerätetyp wie z.B. Festplatten, CD-ROMs oder Streamer gibt es einen speziellen Mid-Level-Treiber.

6.1.1 Maximale Anzahl überprüfter LUNs (»max_scsi_luns=«)

Jedes SCSI-Gerät kann eine Reihe von »Sub-Geräten« enthalten. Das beste Beispiel dafür sind die CD-ROM-Wechsler. Jede CD in diesem Wechsler wird als »Logical Unit Number« (LUN) dieses bestimmten Gerätes angesprochen. Die meisten Geräte jedoch, wie z.B. Festplatten, Bandlaufwerke u.ä. bestehen nur aus einer Geräteeinheit und erhalten LUN Null.

Ein Problem ergibt sich bei einigen Geräten mit einer einzigen LUN. Diese Geräte, alte, aber leider auch neue, besitzen eine schlechte Firmware, die mit der Überprüfung für LUNs ungleich null nicht umgehen kann. Als Reaktion auf die Überprüfung hängen sich die Geräte auf und blockieren im schlimmsten Fall sogar den gesamten SCSI-Bus, so daß auch andere Geräte nicht mehr angesprochen werden können.

Neuere Kernel verfügen über eine Konfigurations-Option, die es ermöglicht, die maximale Anzahl von geprüften LUNs festzulegen. Standardmäßig untersucht man nur LUN Null, um das oben erwähnte Problem zu vermeiden.

Zur Bestimmung der Anzahl von geprüften LUNs beim Booten gibt man `max_scsi_luns=n` als Bootparameter ein, wobei n eine Zahl zwischen 1 und 8 ist. Um oben genannte Probleme zu vermeiden, würde man n=1 verwenden, um solche defekten Geräte nicht zu verwirren.

6.1.2 SCSI Logging (»scsi_logging=«)

Durch Übergabe eines von Null verschiedenen Wertes an diesen Parameter wird das Logging von allen SCSI-Ereignissen (error, scan, mlqueue, mlcomplete, llqueue, llcomplete, hlqueue und hlcomplete) eingeschaltet. Hierbei sollte man beachten, daß über das `/proc/scsi/scsi` Interface besser kontrollieren werden kann, welche Ereignisse protokolliert werden sollen. Diese zweite Lösung kann allerdings nur Ereignisse protokollieren, die nach dem Mounten des `/proc` Dateisystems auftreten.

6.1.3 Parameter für den SCSI-Tape-Treiber (»st=«)

Ein Teil der Boot-Konfiguration des Treibers für SCSI-Bandlaufwerke kann mit folgendem Kommando erfolgen:

```
st=buf_größe[,schreib_grenzwert[,max_bufs]]
```

Die ersten zwei Zahlen werden in kB-Einheiten angegeben. Der Standardwert für `buf_größe` ist 32 kB und die maximal zu bestimmende Größe sind lächerliche 16384 kB. `schreib_grenzwert` ist der Grenzwert, bei dem der Puffer an das Laufwerk weitergegeben wird. Standardwert hierbei sind 30 kB. Die maximale Anzahl von Puffern ändert sich je nach der Zahl der erkannten Laufwerke. Standardwert ist 2. Hier eine Beispielverwendung:

```
st=32,30,2
```

Umfassende Details findet man in der Datei `README.st` im Verzeichnis `scsi` des Kernel-Quell-Baums.

6.2 Parameter für SCSI-Host-Adapter

Allgemeine Anmerkungen zu diesem Abschnitt:

iobase

Der erste I/O Port, der vom SCSI-Host belegt wird. Er wird in der Hexadezimalschreibweise angegeben und liegt für gewöhnlich zwischen `0x200` und `0x3ff`.

irq

Der Hardware-Interrupt, den die Karte per Konfiguration verwendet. Die gültigen Werte sind abhängig von der fraglichen Karte. Normalerweise gelten die Werte 5, 7, 9, 10, 11, 12 und 15. Die anderen Werte werden normalerweise für übliche Peripheriegeräte wie IDE-Festplatten, Disketten, serielle Anschlüsse etc. verwendet.

dma

Der von der Karte verwendete DMA-Kanal (Direct Memory Access). Dies gilt typischerweise nur für Busmaster-Karten. PCI und VLB Karten benötigen keinen ISA-DMA-Kanal.

scsi-id

Die ID, die der Host-Adapter verwendet, um sich auf dem SCSI-Bus zu identifizieren. Nur einige Host-Adapter erlauben die Änderung dieses Werts, da er bei den meisten intern fest eingestellt ist. Gewöhnlich ist der Standardwert ID 7. Die Seagate und Future Domain TMC-950-Boards jedoch verwenden ID 6.

parity

Legt fest, ob der SCSI Host-Adapter von den angeschlossenen Geräten erwartet, daß sie für alle ausgetauschten Informationen einen Paritätswert zur Verfügung stellen. Der Wert 1 aktiviert den Paritäts-Check und 0 schaltet ihn ab. Auch hier gilt, daß nicht alle Adapter die Auswahl eines Paritätsverhaltens als Bootparameter unterstützen.

6.2.1 Adaptec aha151x, aha152x, aic6260, aic6360, SB16-SCSI (»aha152x=«)

Die aha-Zahlen beziehen sich auf Karten und die aic-Zahlen beziehen sich auf den tatsächlichen SCSI-Chip auf diesem Kartentyp, Soundblaster-16 SCSI eingeschlossen.

Die automatische Hardwareerkennung des Treibers für diesen SCSI Host schaut nach einem installierten BIOS. Falls keines vorhanden ist, wird die Karte nicht gefunden. Für diesen Fall wird man einen Bootparameter folgender Form verwenden müssen:

```
aha152x=iobase[,irq[,scsi-id[,reconnect[,parity]]]]
```

Wurde der Treiber mit aktiviertem Debugging kompiliert, dann kann ein sechster Wert zur Bestimmung des Debug-Levels gesetzt werden.

Alle Parameter verhalten sich wie eingangs dieses Abschnitts beschrieben. Ein `reconnect` Wert ungleich null erlaubt Geräten die Verwendung von Disconnect/Reconnect. Hier eine Beispielverwendung:

```
aha152x=0x340,11,7,1
```

Man beachte, daß die Angabe der Parameter einer bestimmten Reihenfolge folgen muß, d.h. wenn man eine Paritätseinstellung angeben will, dann muß man auch einen iobase-, einen irq-, einen scsi-id- und einen reconnect-Wert angeben.

6.2.2 Adaptec aha154x (»aha1542=«)

Dies sind die Karten aus der aha154x-Serie. Die Karten aus der aha1542-Serie verfügen über einen i82077 Floppy-Kontroller, die Karten der Serie aha1540 hingegen nicht. Diese sind Busmaster-Karten und haben Parameter zur Festlegung der »Fairness«, die dazu verwendet wird, den Bus mit anderen Geräten zu teilen. Der Bootparameter sieht folgendermaßen aus:

```
aha1542=iobase[,buson,busoff[,dmaspeed]]
```

Gewöhnlich ist einer der folgenden `iobase` Werte gültig: `0x130`, `0x134`, `0x230`, `0x234`, `0x330` oder `0x334`. Clone-Karten lassen möglicherweise andere Werte zu.

Die `buson, busoff` Werte beziehen sich auf die Anzahl von Mikrosekunden, in denen die Karte den ISA-Bus beherrscht. Die Standardwerte sind 11 μs an und 4 μs aus, so daß andere Karten wie z.B. eine ISA LANCE Ethernetkarte eine Chance haben, auf den ISA-Bus zuzugreifen.

Der Wert `dmaspeed` bezieht sich auf die Geschwindigkeit (in MB/s) in der der DMA-Transfer erfolgt. Der Standardwert ist 5 MB/s. Karten einer neueren Revision ermöglichen die Auswahl dieses Werts als Teil der Konfiguration per Software, ältere Karten verwenden Jumper. Man kann Werte bis 10 MB/s benutzen, vorausgesetzt, das Motherboard kann damit umgehen. Bei der Verwendung von Werten über 5 MB/s sollte man vorsichtig vorgehen.

6.2.3 Adaptec aha274x, aha284x, aic7xxx (»aic7xxx=«)

Diese Boards können einen Parameter folgender Form annehmen:

```
aic7xxx=extended,no_reset
```

Der Wert `extended`, falls ungleich Null, besagt, daß die erweiterte Übersetzung für große Platten eingeschaltet ist. Der Wert `no_reset`, falls ungleich Null, teilt dem Treiber mit, keinen Reset auf dem SCSI-Bus auszuführen, wenn der Host-Adapter beim Booten initialisiert wird.

6.2.4 AdvanSys SCSI Host-Adapter (»advansys=«)

Der AdvanSys-Treiber akzeptiert bis zu 4 I/O Adressen, unter denen der Treiber eine AdvanSys SCSI-Karte sucht. Man beachte, daß diese Werte überhaupt keinen Einfluß auf die EISA- oder PCI-Karten haben. Sie werden nur für die Überprüfung von ISA- und VLB-Karten eingesetzt. Wurde der Treiber mit eingeschaltetem Debugging kompiliert, kann durch Hinzufügen des Parameters `0xdeb[0-f]` bestimmt werden, bis zu welchem Grad Debugging-Meldungen ausgegeben werden sollen. `0-f` ermöglicht die Festlegung des Levels der Debugging-Meldungen auf jeden der 16 Level.

6.2.5 Always IN2000-Host-Adapter (»in2000=«)

Im Gegensatz zu den Bootparametern der anderen SCSI Host-Adapter verwendet der IN2000-Treiber für die meisten seiner Integer-Parameter ASCII-Strings als Präfix. Hier eine Liste von unterstützten Parametern:

ioport:addr

Wobei `addr` die I/O-Adresse einer Karte ist, die gewöhnlich kein ROM besitzt.

noreset

Verhindert einen Reset des SCSI-Busses beim Booten. Diesem Parameter können keine optionalen Argumente übergeben werden.

nosync:x

`x` ist eine Bitmaske, wobei die ersten 7 Bit mit den 7 möglichen SCSI-Geräten übereinstimmen (Bit 0 für das Gerät mit ID0, etc). Um die Sync-Übertragung auf ein Gerät zu *verhindern*, setzt man dessen Bit. Standardmäßig ist Sync für alle Geräte ausgeschaltet.

period:ns

`ns` ist die minimale # von Nanosekunden für einen SCSI-Datentransfer. Standard ist 500; erlaubte Werte sind 250 bis 1000.

disconnect:x

`x = 0` verhindert grundsätzlich Disconnects; `x = 2` erlaubt immer Disconnects. `x = 1` führt »adaptive« Disconnects durch. Dieses ist der Standard und wohl allgemein die beste Wahl.

debug:x

Ist »DEBUGGING_ON« angegeben, dann ist `x` eine Bitmaske, durch die verschiedene Arten von Debug-Ausgaben ein- oder ausgeschaltet werden; siehe hierzu die Definition der `DB_xxx` Konstanten in `in2000.h`.

proc:x

Ist »PROC_INTERFACE« angegeben, dann ist `x` eine Bitmaske, die festlegt, wie die `/proc`-Schnittstelle funktioniert und was sie macht; siehe hierzu die Definition der `PR_xxx` Konstanten in `in2000.h`.

Hier einige Anwendungs-Beispiele :

```
in2000=ioport:0x220,noreset
in2000=period:250,disconnect:2,nosync:0x03
in2000=debug:0x1e
in2000=proc:3
```

6.2.6 AMD AM53C974-basierte Hardware (»AM53C974=«)

Im Gegensatz zu anderen Treibern verwendet dieser keine Bootparameter, um I/O-, IRQ- oder DMA-Kanäle festzulegen. Da der AM53C974 ein PCI-Gerät ist, sollte dafür auch kein Grund bestehen. Statt dessen teilen die Parameter für gewöhnlich die Transfermodi und Transferraten mit, die zwischen dem Host- und dem Zielgerät verwendet werden sollen. Dies läßt sich am besten anhand eines Beispiels beschreiben:

```
AM53C974=7,2,8,15
```

Dies würde folgendermaßen interpretiert werden: Für die Kommunikation zwischen dem SCSI-Kontroller mit ID7 und dem SCSI-Gerät mit ID2 soll eine Transferrate von 8 MHz im Synchronmodus mit max. 15 Bytes Offset ausgehandelt werden. Weitere Informationen findet man in der Datei `linux/drivers/scsi/README.AM53C974`

6.2.7 BusLogic SCSI-Hosts mit v1.2.x Kerneln (»buslogic=«)

In älteren Kerneln akzeptiert der buslogic-Treiber nur einen Parameter, und zwar die iobase. Folgende Werte sind zulässig: `0x130`, `0x134`, `0x230`, `0x234`, `0x330` und `0x334`.

6.2.8 BusLogic SCSI-Hosts mit v2.x Kerneln (»BusLogic=«)

Bei v2.x Kerneln akzeptiert der BusLogic-Treiber viele Parameter. Man beachte die neue Schreibweise des Parameters im Vergleich zu den alten Kernels. Dieser Parameter kennt zu viele Option, als das man diese hier alle auflisten könnte. Eine komplette Beschreibung der Optionen ist in der Datei `linux/drivers/scsi/BusLogic.c` zu finden, wenn man nach der Zeichenkette `BusLogic=` sucht.

6.2.9 EATA SCSI-Karten (»eata=«)

Seit den späten Kernelversionen v2.0 akzeptieren die EATA-Treiber die Überprüfung eines Bootparameters, der die iobase(s) bestimmt. Dieses sieht folgendermaßen aus:

```
eata=iobase1[,iobase2][,iobase3]...[,iobaseN]
```

Der Treiber überprüft die Adressen in der Reihenfolge, in der sie aufgelistet sind.

6.2.10 Future Domain TMC-8xx, TMC-950 (»tmc8xx=«)

Der Überprüfungscode für diese SCSI-Hosts schaut nach einem installierten BIOS; falls keines vorhanden ist, wird die Karte nicht gefunden werden. Auch wenn der Signatur-String des BIOS nicht erkannt wurde, wird die Karte nicht gefunden. In jedem der beiden Fälle wird man dann einen Bootparameter folgender Form verwenden:

```
tmc8xx=mem_base,irq
```

Der Wert `mem_base` ist der Wert des in den Speicher eingeblendeten I/O-Bereichs, den die Karte verwendet. Dieser ist für gewöhnlich einer der folgenden Werte: `0xc8000`, `0xca000`, `0xcc000`, `0xce000`, `0xdc000` oder `0xde000`.

6.2.11 Future Domain TMC-16xx, TMC-3260, AHA-2920 (»fdomain=«)

Der Treiber erkennt diese Karten entsprechend einer Liste von bekannten BIOS ROM-Signaturen. Eine komplette Liste bekannter BIOS-Ausgaben erhält man in der Datei `linux/drivers/scsi/fdomain.c`. Diese wird mit einer Fülle von Informationen eingeleitet. Ist das BIOS dem Treiber nicht bekannt, dann kann man dies folgendermaßen überbrücken:

```
fdomain=iobase,irq[,scsi_id]
```

6.2.12 IOMEGA Parallel Port ZIP-Laufwerk (»ppa=«)

Dieses ist ein Treiber für den IOMEGA Parallel Port SCSI Adapter, welcher in den IOMEGA ZIP-Laufwerken enthalten ist. Er könnte auch mit dem original IOMEGA PPA3-Gerät funktionieren. Der Bootparameter für diesen Treiber sieht folgendermaßen aus:

```
ppa=iobase,speed_high,speed_low,nybble
```

Alle außer `iobase` sind optional festlegbare Werte. Will man einen der optionalen Parameter verändern, dann sollte man einen Blick in die Datei `linux/drivers/scsi/README.ppa` werfen. Dort findet man genaue Informationen darüber, was von diesen Parametern kontrolliert wird.

6.2.13 NCR5380-basierte Kontroller (»ncr5380=«)

Je nach Board kann der 5380 entweder über I/O-Ports oder einen eingeblendeten Speicherbereich angesprochen werden. Eine Adresse unter 0x400 ist für gewöhnlich ein I/O-Port. PCI- und EISA-Hardware verwenden jedoch I/O-Adressen über 0x3ff. In jedem Fall gibt man die Adresse, den Wert des Interrupts und des DMA-Kanals an. Hier ein Beispiel einer I/O-Karte: `ncr5380=0x350,5,3`. Verwendet die Karte keine Interrupts, dann können mit einem IRQ-Wert von 255 (`0xff`) Interrupts deaktiviert werden. Ein IRQ-Wert von 254 bedeutet eine automatische Hardwareerkennung. Weitere Informationen findet man in der Datei `linux/drivers/scsi/README.g_NCR5380`.

6.2.14 NCR53c400-basierte Kontroller (»ncr53c400=«)

Die generische 53c400-Unterstützung erfolgt mit demselben Treiber wie für die oben erwähnte generische 5380-Unterstützung. Der Bootparameter ist identisch zum obig genannten, mit der Ausnahme, daß vom 53c400 kein DMA-Kanal verwendet wird.

6.2.15 NCR53c406a-basierte Kontroller (»ncr53c406a=«)

Dieser Treiber verwendet einen Bootparameter folgender Form

```
ncr53c406a=PORTBASE,IRQ,FASTPIO
```

wobei die `IRQ`- und `FASTPIO`-Parameter optional sind. Ein Interrupt-Wert von 0 schaltet die Verwendung von Interrupts aus. Der Wert 1 für den `FASTPIO`-Parameter aktiviert die Verwendung von `insl`- und `outsl`-Anweisungen statt den aus einem einzigen Byte bestehenden `inb`- und `outb`-Anweisungen. Während der Kompilierung eines neuen Kernels steht DMA als Option zur Verfügung.

6.2.16 Pro Audio Spectrum (»pas16=«)

PAS16 verwendet einen NCR5380 SCSI-Chip und neuere Modelle unterstützen die Konfiguration ohne Jumper. Der Bootparameter sieht folgendermaßen aus:

```
pas16=iobase,irq
```

Der einzige Unterschied besteht darin, daß man einen IRQ-Wert von 255 verwenden kann, welcher dem Treiber mitteilt, ohne die Verwendung von Interrupts zu arbeiten, was jedoch Geschwindigkeitsverluste mit sich bringt. Die `iobase` ist gewöhnlich `0x388`.

6.2.17 Seagate ST-0x (»st0x=«)

Der Code zur Überprüfung für diese SCSI-Hosts sucht nach einem installierten BIOS. Ist keines vorhanden, wird die Karte nicht gefunden. Auch wenn die Signatur-Zeichenkette des BIOS nicht erkannt wird, wird die Karte nicht gefunden. In jedem Fall wird man einen Bootparameter folgender Form verwenden:

```
st0x=mem_base,irq
```

Der Wert `mem_base` ist der Wert der in den Speicher eingeblendeten I/O-Region, den die Karte verwendet. Dieser wird für gewöhnlich einer der folgenden Werte sein: `0xc8000`, `0xca000`, `0xcc000`, `0xce000`, `0xdc000` oder `0xde000`.

6.2.18 Trantor T128 (»t128=«)

Diese Karten basieren ebenfalls auf dem NCR5380-Chip und verstehen folgende Optionen:

```
t128=mem_base,irq
```

Gültige Werte für `mem_base` sind: `0xcc000`, `0xc8000`, `0xdc000` und `0xd8000`.

6.2.19 Ultrastor SCSI-Karten (»u14-34f=«)

Man beachte, daß es anscheinend zwei unabhängige Treiber für diese Karte gibt, nämlich `CONFIG_SCSI_U14_34F`, der `u14-34f.c` benutzt, und `CONFIG_SCSI_ULTRASTOR`, der `ultrastor.c` verwendet. u14-34f ist derjenige, der seit den späten Kernelversionen v2.0 einen Bootparameter folgender Form akzeptiert:

```
u14-34f=iobase1[,iobase2][,iobase3]...[,iobaseN]
```

Der Treiber überprüft die Adressen in der Reihenfolge, wie sie aufgelistet sind.

6.2.20 Western Digital WD7000-Karten (»wd7000=«)

Die automatische Erkennung für wd7000 sucht nach einer bekannten BIOS ROM-Zeichenkette und kennt einige Standardeinstellungen. Werden die korrekten Werte für die eigene Karte nicht geliefert oder hat man eine nicht erkannte BIOS-Version, dann kann man einen Bootparameter folgender Form verwenden:

```
wd7000=irq,dma,iobase
```

6.3 SCSI-Host-Adapter, die keine Bootparameter akzeptieren

Zur Zeit verwenden die Treiber folgender SCSI-Karten keine Bootparameter. In einigen Fällen kann man Werte »hartkodieren«, indem man, falls nötig, den Treiber selbst editiert.

- Adaptec aha1740 (EISA-Überprüfung)
- NCR53c7xx,8xx (PCI, beide Treiber)
- Qlogic Fast (0x230, 0x330)
- Qlogic ISP (PCI)

7 Festplatten

In diesem Abschnitt werden alle Bootparameter aufgelistet, die mit Standard-MFM/RLL-, ST-506-, XT- und (E)IDE-Laufwerken verbunden sind. Man beachte, daß sowohl der IDE- als auch der generische ST-506 HD-Treiber die Option `hd=` akzeptieren.

7.1 IDE Disk-/CD-ROM-Treiber-Parameter

Der IDE-Treiber akzeptiert eine Reihe von Parametern, die von Spezifikationen der Plattengeometrie bis zur Unterstützung für höher entwickelte oder kaputte Kontroller-Chips reichen. Es folgt eine Zusammenfassung aller möglichen Bootparameter. Benötigt man weitere Informationen, sollte man *unbedingt* einen Blick in die Datei `ide.txt` im Verzeichnis `linux/Documentation` werfen, dem diese Zusammenfassung entnommen wurde.

Bei dem Parameter `hdx=` können für `x` Werte von `a` bis `h` verwendet werden. Für `x` bei dem Parameter `idex=` können Werte von `0` bis `3` Verwendung finden. Beispielsweise würde Linux `hdc=` und `ide1=` erkennen.

hdx=noprobe
: Laufwerk mag vorhanden sein, aber es soll nicht automatisch nach ihm gesucht werden.

hdx=none
: Das Laufwerk ist *nicht* vorhanden. Die CMOS-Einstellungen sollen ignoriert und das Laufwerk nicht erkannt werden.

hdx=nowerr
: Das WRERR_STAT-Bit soll bei diesem Laufwerk ignoriert werden.

hdx=cdrom
: Das Laufwerk ist vorhanden und es handelt sich um ein CD-ROM-Laufwerk.

hdx=cyl,head,sect
: Ein Plattenlaufwerk mit angegebener Geometrie ist vorhanden.

hdx=autotune
: Der Treiber soll versuchen, die Geschwindigkeit der Schnittstelle auf den schnellsten unterstützten PIO-Modus einzustellen. Wenn möglich, soll nur der Modus dieses Laufwerkes verändert werden. Diese Option wird nicht von allen Chipsätzen voll unterstützt. Kann zu Problemen mit bestimmten IDE-Laufwerken führen.

idex=noprobe

Es soll nicht versucht werden, auf diese Schnittstelle zuzugreifen oder sie zu verwenden.

idex=base

Die IDE-Schnittstelle soll unter der angegebenen Adresse gesucht werden, wobei `base` normalerweise `0x1f0` oder `0x170` ist und angenommen wird, daß `ctl` `base`+0x206 ist.

idex=base,ctl

Es wird sowohl `base` als auch `ctl` bestimmt.

idex=base,ctl,irq

Legt `base`, `ctl` und `irq` fest.

idex=autotune

Hat die gleiche Funktion wie `hdx=autotune`, bezieht sich allerdings auf alle Laufwerke an einer Schnittstelle.

idex=noautotune

Der Treiber versucht *nicht*, die Geschwindigkeit der Schnittstelle einzustellen. Dies ist der Standard für die meisten Chipsätze mit Ausnahme des cmd640.

idex=serialize

Es wird *nicht* gleichzeitig auf `idex` und eine weitere IDE-Schnittstelle zugegriffen. Dies ist für einige fehlerhafte Chipsätze sehr nützlich.

Das Folgende gilt *nur* auf `ide0`, und die Standardwerte für die `base` und `ctl` Ports dürfen *nicht* geändert werden.

ide0=dtc2278

Prüfe und unterstütze die DTC2278-Schnittstelle.

ide0=ht6560b

Prüfe und unterstütze die HT6560B-Schnittstelle.

ide0=cmd640_vlb

Ist nur für VLB-Karten mit dem CMD640-Chip notwendig. Die PCI-Version wird automatisch erkannt.

ide0=qd6580

Prüfe und unterstütze die qd6580-Schnittstelle.

ide0=ali14xx

Prüfe und unterstütze die ali14xx-Chipsätze (ALI M1439/M1445).

ide0=umc8672

Prüfe und unterstütze die umc8672-Chipsätze.

Alles Übrige wird mit der Nachricht »BAD OPTION« abgewiesen.

7.2 Standard ST-506-Platten-Treiber-Optionen (»hd=«)

Der Standard-Plattentreiber kann, ähnlich wie der IDE-Treiber, Parameter für die Geometrie von Platten akzeptieren. Man beachte jedoch, daß er nur drei Werte (C/H/S) erwartet; nur einer mehr oder weniger, und der Parameter wird einfach von ihm ignoriert. Er akzeptiert auch nur `hd=` als Argument, d.h. `hda=`, `hdb=` usw. sind hier nicht gültig. Das Format sieht folgendermaßen aus:

```
hd=cyls,heads,sects
```

Wenn zwei Platten installiert sind, wird das Obenstehende mit den Geometrie-Parametern der zweiten Platte wiederholt.

7.3 XT-Platten-Treiber-Optionen (»xd=«)

Sollten Sie zu den Unglücklichen gehören, die eine dieser alten 8 Bit-Karten verwenden, die Daten keuchend mit 125 kB/s verschieben, dann kommt hier der Knüller. Der Überprüfungscode für diese Karten schaut nach einem installierten BIOS. Falls keines vorhanden ist, dann wird die Karte nicht erkannt werden. Wenn der Signatur-String Ihres BIOS nicht erkannt wurde, wird sie ebenfalls nicht gefunden. In jedem Fall wird man dann einen Bootparameter folgender Form verwenden müssen:

```
xd=type,irq,iobase,dma_chan
```

Der Wert `type` bestimmt den einzelnen Hersteller der Karte, und zwar wie folgt:

- `0`: generic
- `1`; DTC
- `2,3,4`: Western Digital
- `5,6,7`: Seagate
- `8`: OMTI

Der einzige Unterschied zwischen den verschiedenen Typen desselben Herstellers liegt in dem BIOS-String, der für die Erkennung verwendet wird. Dieser wird nicht benutzt, wenn der Typ angegeben wurde.

Die `xd_setup()`-Funktion überprüft die Werte nicht und nimmt an, daß alle vier Werte eingetragen wurden. Man sollte sie nicht enttäuschen. Hier ist eine beispielhafte Verwendung für einen WD1002-Kontroller mit ausgeschaltetem bzw. entferntem BIOS, wobei die standardmäßigen Parameter vom XT-Kontroller verwendet werden:

```
xd=2,5,0x320,3
```

8 CD-ROMs (nicht-SCSI/-ATAPI/-IDE)

In diesem Abschnitt werden alle möglichen Bootparameter aufgelistet, die zu den CD-ROM-Geräten gehören. Man beachte, daß dies *nicht* für SCSI- oder IDE-/ATAPI-CD-ROMs gilt. Informationen über diese CD-ROM-Typen findet man in den entsprechenden Abschnitten.

Man beachte, daß es für die meisten dieser CD-ROM-Laufwerke Dokumentationsdateien gibt, die man lesen sollte. Sie alle sind günstig plaziert: `linux/Documentation/cdrom`.

8.1 Die Aztech-Schnittstelle (»aztcd=«)

Dieser Kartentyp hat folgende Syntax:

```
aztcd=iobase[,magic_number]
```

Setzt man `magic_number` auf `0x79`, dann wird der Treiber einen Versuch starten und im Falle einer unbekannten Firmware irgendwie laufen. Alle übrigen Werte werden ignoriert.

8.2 Die CDU-31A- und CDU-33A-Sony-Schnittstelle (»cdu31a=«)

Diese CD-ROM-Schnittstelle kann auf einigen der Pro Audio Spectrum Soundkarten gefunden werden und auf anderen Schnittstellen-Karten von Sony. Die Syntax ist folgendermaßen:

```
cdu31a=iobase,[irq[,is_pas_card]]
```

Setzt man einen IRQ-Wert auf Null, dann wird dem Treiber damit mitgeteilt, daß die Hardware Interrupts nicht unterstützt, was bei einigen PAS-Karten der Fall ist. Unterstützt die eigene Karte Interrupts, dann sollte man diese nutzen, da diese die CPU-Last des Treibers verringern.

Falls eine Pro Audio Spectrum verwendet werden soll, muß für `is_pas_card` die Zeichenkette `PAS` eingetragen werden. Andernfalls sollte die Option überhaupt nicht bestimmt werden.

8.3 Die CDU-535-Sony-Schnittstelle (»sonycd535=«)

Diese CD-ROM-Schnittstelle hat folgende Syntax:

```
sonycd535=iobase[,irq]
```

Für die `iobase` kann eine Null als »Platzhalter« verwendet werden, falls man einen IRQ-Wert bestimmen will.

8.4 Die GoldStar-Schnittstelle (»gscd=«)

Diese CD-ROM-Schnittstelle hat folgende Syntax:

```
gscd=iobase
```

8.5 Die ISP16-Schnittstelle (»isp16=«)

Diese CD-ROM-Schnittstelle hat folgende Syntax:

```
isp16=[port[,irq[,dma]]][[,]drive_type]
```

Der Wert Null für `irq` oder `dma` besagt, daß sie nicht benutzt werden. Erlaubte Werte für `drive_type` sind `noisp16`, `Sanyo`, `Panasonic`, `Sony` und `Mitsumi`. Die Verwendung von `noisp16` deaktiviert den gesamten Treiber.

8.6 Die Mitsumi Standard-Schnittstelle (»mcd=«)

Diese CD-ROM-Schnittstelle hat folgende Syntax:

```
mcd=iobase,[irq[,wait_value]]
```

`wait_value` wird als interner Timeout-Wert für diejenigen verwendet, die Probleme mit ihrem Laufwerk haben. Ob diese Option vorhanden ist, kann während der Kompilierung des Kernels festgelegt werden.

8.7 Die Mitsumi XA-/MultiSession-Schnittstelle (»mcdx=«)

Zur Zeit verfügt dieser experimentelle Treiber über eine Setup-Funktion, jedoch sind bis jetzt (seit 1.3.15) keine Parameter implementiert. Dieser Treiber ist für die gleichen Laufwerke wie der vorher beschriebene gedacht, allerdings bietet dieser weitere Möglichkeiten.

8.8 Die Optics Storage-Schnittstelle (»optcd=«)

Dieser Kartentyp hat folgende Syntax:

```
optcd=iobase
```

8.9 Die Philips CM206-Schnittstelle (»cm206=«)

Dieser Kartentyp hat folgende Syntax:

```
cm206=[iobase][,irq]
```

Zahlen zwischen 3 und 11 werden vom Treiber als IRQ-Werte interpretiert und Zahlen zwischen `0x300` und `0x370` als I/O Ports, so daß man eine oder beide Zahlen in beliebiger Reihenfolge angeben kann. Der Treiber akzeptiert auch `cm206=auto` zum Aktivieren der automatischen Hardwareerkennung.

8.10 Die Sanyo-Schnittstelle (»sjcd=«)

Diese Karte hat folgende Syntax:

```
sjcd=iobase[,irq[,dma_channel]]
```

8.11 Die SoundBlaster Pro-Schnittstelle (»sbpcd=«)

Diese Karte hat folgende Syntax

```
sbpcd=iobase,type
```

wobei `type` einer der folgenden Zeichenketten ist: `SoundBlaster`, `LaserMate` oder `SPEA`. Groß- oder Kleinschreibung spielt keine Rolle. Die I/O-Basisadresse ist die der CD-ROM-Schnittstelle und *nicht* die des Teils der Karte, der den Sound produziert.

9 Serielle Schnittstellen und ISDN

9.1 Der ICN ISDN-Treiber (»icn=«)

Dieser ISDN-Treiber erwartet einen Bootparameter folgender Form:

```
icn=iobase,membase,icn_id1,icn_id2
```

Hierbei ist `iobase` die Adresse des I/O Ports der Karte und. `membase` die Startadresse des Shared Memory Bereiches. Die beiden `icn_id` Werte sind einmalige ASCII-Zeichenketten zur Identifizierung der Karte.

9.2 Der PCBIT ISDN-Treiber (»pcbit=«)

Dieser Bootparameter verwendet Paare von Ganzzahlen-Argumenten, z.B.:

```
pcbit=membase1,irq1[,membase2,irq2]
```

Hierbei ist `membaseN` die Shared Memory-Basisadresse und `irqN` der Interrup der Nten Karte. Standard ist IRQ 5 und ein eingeblendeter Speicher ab Adresse `0xD0000`.

9.3 Der Teles ISDN-Treiber (»teles=«)

Dieser ISDN-Treiber erwartet einen Bootparameter folgender Form:

```
teles=iobase,irq,membase,protocol,teles_id
```

`iobase` ist hierbei die I/O Port Adresse und `membase` die Shared Memory-Basisadresse der Karte; `irq` ist der von der Karte verwendete Interrupt und `teles_id` ist ein eindeutiger ASCII-String-Bezeichner.

9.4 Der DigiBoard-Treiber (»digi=«)

Der DigiBoard-Treiber akzeptiert eine Zeichenkette von sechs durch Kommas getrennten Bezeichnern oder Ganzzahlen. Hier die sechs Werte in Reihenfolge:

1. Aktivieren (`E`) oder Deaktivieren (`D`) der Karte
2. Kartentyp: PC/Xi (`0`), PC/Xe (`1`), PC/Xeve (`2`) oder PC/Xem (`3`)
3. Aktivieren (`E`) oder Deaktivieren (`D`) der wechselnden Pin-Anordnung
4. Anzahl der I/O-Ports dieser Karte
5. I/O-Port, über den die Karte konfiguriert wird (hexadezimal, falls String-Bezeichner verwendet werden)
6. Basisadresse des Speicher-Fensters (hexadezimal, falls String-Bezeichner verwendet werden)

Hier ein Beispiel eines korrekten Bootparameters sowohl in Bezeichner- als auch in Ganzzahlen-Form:

```
digi=E,PC/Xi,D,16,200,D0000
digi=1,0,0,16,512,851968
```

Man beachte, daß der Treiber standardmäßig den I/O-Port `0x200` und die Shared Memory-Basisadresse `0xD0000` voreingestellt hat, falls kein `digi=` Bootparameter angegeben wurde. Es wird keine automatische Hardwareerkennung durchgeführt. Weitere Informationen findet man in der Datei `linux/Documentation/digiboard.txt`.

9.5 Der RISCom/8 Multiport Seriell-Treiber (»riscom8=«)

Bis zu vier Karten werden unterstützt, indem man vier eindeutige I/O Port-Werte für jede einzelne Karte angibt. Weitere Informationen findet man in der Datei `linux/Documentation/riscom8.txt`.

9.6 Das Baycom Seriell/Parallel Radio Modem (»baycom=«)

Der Bootparameter für diese Geräte hat folgendes Format:

```
baycom=modem,io,irq,options[,modem,io,irq,options]
```

Die Verwendung von `modem`=1 bedeutet, daß man das ser12-Gerät hat, `modem`=2 bedeutet, man hat das par96-Gerät. `options`=0 bedeutet die Verwendung von Hardware-DCD, und `options`=1 bedeutet die Verwendung von Software-DCD. `io` und `irq` sind wie gewöhnlich die I/O Port-Basisadresse und der Interrupt. Weitere Informationen findet man in der Datei `README.baycom`, die sich zur Zeit im Verzeichnis `/linux/drivers/char/` befindet.

10 Andere Hardware-Geräte

Alle übrigen Geräte, die nicht in eine der oben erwähnten Kategorien passen, wurden hier zusammengefaßt.

10.1 Ethernet-Geräte (»ether=«)

Unterschiedliche Treiber verwenden unterschiedliche Parameter. Ihnen allen gemeinsam ist, daß sie alle über einen IRQ-Wert, einen Wert der I/O Port-Basisadresse und einen Namen verfügen. In seiner allgemeinsten Form schaut dies in etwa so aus:

```
ether=irq,iobase[,param_1[,param_2,...param_8]]],name
```

Das erste, nicht numerische Argument wird als Name verwendet. Die `param_n` Werte haben normalerweise für jede(n) einzelne Karte bzw. Treiber eine unterschiedliche Bedeutung. Typische `param_n` Werte werden zur Bestimmung von Dingen wie der »Shared Memory«-Adresse, der Auswahl der Schnittstelle, der Bestimmung des DMA-Kanal u.ä. verwendet.

Dieser Parameter wird am häufigsten dafür eingesetzt, die automatische Hardwareerkennung für eine zweite Ethernetkarte zu erzwingen, da standardmäßig nur nach einer gesucht wird. Dieses erreicht man mit einem einfachen Befehl:

```
ether=0,0,eth1
```

Man beachte, daß im obigen Beispiel der Wert Null für den Interrupt und die I/O-Basisadresse den oder die Treiber auffordert, eine automatische Hardwareerkennung durchzuführen.

Falls Sie Module verwenden, sollten sie folgendes bedenken: Oben genanntes Kommando wird *keine* Überprüfung nach einer zweiten Karte erzwingen, wenn der Treiber für diese Karte zur Laufzeit als Modul geladen wird, weil er nicht im Kernel einkompiliert ist. Die meisten Linux-Distributionen verwenden ein bloßes Kernel-Gerüst zusammen mit einer großen Auswahl an Treibern in Modulform.

Das *Ethernet HOWTO* verfügt über eine komplette und ausführliche Dokumentation über die Verwendung mehrerer Karten und über die spezifische Implementation der `param_n` Werte bei den jeweiligen Treibern. Interessierten Lesern sei empfohlen, sich in dem entsprechenden Abschnitt dieses Dokuments komplette Informationen über ihre spezielle Karte zu holen.

10.2 Der Disketten-Treiber (»floppy=«)

Es gibt eine Fülle von Optionen für den Treiber der Diskettenlaufwerke, die in der Datei `README.fd` unter `linux/drivers/block` aufgelistet sind. Diese Datei enthält zu viele Optionen, um sie hier alle näher zu beschreiben. Statt dessen wird nur auf die Optionen eingegangen, die eventuell während der Installation von Linux auf Systemen mit nicht ganz alltäglicher Hardware benötigt werden.

floppy=0,daring

Teilt dem Disketten-Treiber mit, daß der Disketten-Controller mit Vorsicht behandelt werden soll.

floppy=thinkpad

Teilt dem Disketten-Treiber mit, daß ein Thinkpad verwendet wird. Thinkpads verwenden eine umgekehrte Konvention für die Leitung, die einen Wechsel der Diskette anzeigt.

floppy=nodma

Mit dieser Option wird dem Disketten-Treiber mitgeteilt, daß für Datentransfers kein DMA verwendet werden soll. Dieses wird für HP Omnibooks benötigt, die keinen funktionsfähigen DMA-Kanal für den Disketten-Treiber besitzen. Außerdem ist diese Option hilfreich, wenn man regelmäßig die Meldung »unable to allocate DMA memory« erhält. Von der Verwendung dieser Option muß abgeraten werden, falls ein Disketten-Controller ohne einen FIFO zum Einsatz kommt. Hierzu zählen z.B. die Typen 8272A und 82072; 82072A und spätere Versionen haben hingegen einen FIFO. Der Typ des Controllers wird beim Bootvorgang angezeigt. Außerdem wird wenigstens ein 486er benötigt, um diese Option verwenden zu können.

floppy=broken_dcl

Die Leitung, die einen Diskettenwechsel anzeigt, soll nicht verwendet werden. Statt dessen wird angenommen, daß die Diskette gewechselt wurde, wenn das Device des Laufwerkes neu geöffnet wird. Diese Option wird bei einigen Systemen benötigt, bei den diese Leitung nicht funktioniert oder nicht vorhanden ist.

Diese Option sollte jedoch als Notlösung angesehen werden, da hierdurch Zugriffe auf das Diskettenlaufwerk nicht zu effizient durchgeführt werden können, da der Cache unnötigerweise geleert werden muß. Auch sind Zugriffe dann nicht ganz so zuverlässig. Falls irgendwelche Probleme mit der DCL (Disk Change Line) auftauchen, sollte man zuerst immer die Kabelverbindung und die Stellung der entsprechenden Jumper überprüfen. Allerdings sind einige ältere Laufwerke und Notebooks dafür bekannt, daß sie keine DCL besitzen.

floppy=debug

Durch diese Option werden einige zusätzliche Meldungen vom Kernel ausgegeben, die bei der Fehlersuche nützlich sein können.

floppy=messages

Mittels dieser Option werden Meldungen von Kernel über bestimmte Operationen ausgegeben: Wechsel der Diskette, Überläufe und automatische Erkennung.

10.3 Der Sound-Treiber (»sound=«)

Der Treiber für Soundkarten kann auch Bootparameter annehmen, um die hineinkompilierten Werte zu übergehen. Dies wird jedoch nicht empfohlen, da es ziemlich komplex ist und die Dokumentation hierzu im Kernel auf mysteriöse Weise verschwunden ist. Statt dessen sollten die Treiber für die Soundkarten besser als Modul genutzt werden oder der Kernel mit den eigenen Werten neu übersetzt werden.

Falls Sie trotzdem alle dem diesen Parameter benutzen möchten, der vom Kernel in der Datei `dev_table.c` im Verzeichnis `linux/drivers/sound` verarbeitet wird, so sieht der Syntax des Parameters so aus:

```
sound=device1[,device2[,device3...[,device11]]]
```

Hierbei ist jeder `deviceN` Wert von folgendem Format ist: `0xDTaaaId`. Die Bytes werden folgendermaßen verwendet:

- `D`: zweiter DMA-Kanal (Null, wenn nicht vorhanden)
- `T`: Geräte-Typ: 1=FM, 2=SB, 3=PAS, 4=GUS, 5=MPU401, 6=SB16, 7=SB16-MIDI usw. Eine komplette Liste der ersten 26 Typen ist in der Datei `linux/include/linux/soundcard.h` zu finden. Die weiteren Typen sind in der Datei `linux/drivers/sound/dev_table.h` nachzulesen. Bitte vergessen Sie nicht, diese Werte in Hexadezimalzahlen umzuwandeln.
- `aaa`: I/O-Adresse als Hexadezimalzahl
- `I`: Interrupt als Hexadezimalzahl (i.e 10=a, 11=b, ...)
- `d`: erster DMA-Kanal

Wie man sieht, ein ganz schönes Durcheinander. Man ist wohl besser beraten, sich, wie empfohlen, Module zu verwenden oder seine eigenen, persönlichen Werte hineinzukompilieren. Die Verwendung des Bootparameters `sound=0` deaktiviert den gesamten Sound-Treiber.

10.4 Der Bus-Maus-Treiber (»bmouse=«)

Der Bus-Maus-Treiber akzeptiert nur einen Parameter, und zwar den Wert des zu verwendenden Interrupts.

10.5 Der MS-Bus-Maus-Treiber (»msmouse=«)

Der MS-Maustreiber akzeptiert nur einen Parameter, und zwar den zu verwendenden Hardware IRQ-Wert.

10.6 Der Druckertreiber (»lp=«)

Mit diesem Parameter kann dem Drucktreiber mitteilen werden, welche Ports verwendet werden sollen und welche nicht. Letzteres ist dann praktisch, wenn man verhindern will, daß der Druckertreiber alle zur Verfügung stehenden parallelen Ports beansprucht, so daß andere Treiber wie z.B. PLIP oder PPA sie statt dessen verwenden können.

Das Argument besteht aus mehreren Paaren von I/O-Adressen und Interrupts. `lp=0x3bc,0,0x378,7` verwendet z.B. den Port auf 0x3bc im IRQ-losen Polling-Modus, und benutzt IRQ 7 für den Port auf 0x378. Der Port unter 0x278 würde, falls vorhanden, nicht überprüft werden, da die automatische Hardwareerkennung nur ohne ein `lp=`-Argument stattfindet. Zum kompletten Deaktivieren des Druckertreibers kann man `lp=0` verwenden.

11 Schlußbemerkung

Sollten Ihnen irgendwelche Tippfehler ins Auge gesprungen sein, oder haben Sie veraltete Informationen in diesem Dokument gefunden, dann lassen Sie es mich bitte wissen. Es kann schnell mal passieren, daß etwas übersehen wurde, da der Kernel im Laufe der Jahre doch stark gewachsen ist.

Danke,

Paul Gortmaker (`p_gortmaker@yahoo.com`)