Linux BootPrompt HOWTO

Paul Gortmaker (Paul.Gortmaker@anu.edu.au) und Antje Faber (af@caldera.de) v1.13, 15 November 1997

Dies ist das BootPrompt-Howto, welches eine Zusammenstellung aller möglichen Bootparameter ist, 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.

Inhalt

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 andernfalls 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.

WICHTIGER HINWEIS FÜR DIE VERWENDUNG VON MODULEN: Ein Kennzeichen von Bootprompt-Parametern ist die ausschließliche Anwendung auf Hardware-Treiber, die direkt in den Kernel kompiliert werden. Sie haben *keine Auswirkung* auf Treiber, die als Module geladen sind. Die meisten Distributionen verwenden Module. Sollten Zweifel bestehen, sollte man `man depmod` und `man modprobe` zusammen mit `/etc/conf.modules` anschauen.

Die derzeitige Ausgabe beinhaltet alle Kernel bis einschließlich v2.0.31. Es werden ebenfalls einige Eigenschaften dokumentiert, die einzigartig für Entwicklungs-/Test- Kernels bis v2.1.6x sind.

Das *BootPrompt HOWTO* wurde geschrieben und wird gepflegt von:

Paul Gortmaker (`Paul.Gortmaker@anu.edu.au`)

(Man beachte, daß die speziellen Bootprompt-Parameter für nicht-i386-Ports und -Geräte (speziell Atari/Amiga) zur Zeit undokumentiert sind.)

1.1 Ablehnungshinweise und Copyright

Dieses Dokument ist *kein* Evangelium. Es bietet jedoch möglicherweise die aktuellste Information, die man finden kann. Man ist selbst dafür verantwortlich, was mit der eigenen Hardware geschieht. Falls die Hardware in Rauch und Flammen aufgeht (...nahezu unmöglich!) übernehme ich dafür keine Verantwortung. DER AUTOR UND DIE ÜBERSETZER ÜBERNEHMEN KEINE HAFTUNG FÜR EVENTULLE FOLGESCHÄDEN AUFGRUND DER ANWENDUNG DER IN DIESEM DOKUMENT BEREITGESTELLTEN INFORMATIONEN.

Falls Sie beabsichtigen, dieses Dokument in eine Veröffentlichung aufzunehmen, nehmen Sie bitte Kontakt zu 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 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.

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. Das Verzeichnis `linux` findet sich normalerweise unter `/usr/src/`. 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 werfen, die mit der Software zum Booten des Kernels ausgeliefert wird (z.B. LILO oder loadlin). Nachfolgend wird ein kurzer Überblick gegeben, der jedoch keinen Ersatz für die mit der Bootsoftware ausgelieferte Dokumentation darstellt.

1.3 Linux Newsgruppen

Bei Fragen über die Weitergabe von Bootparametern an den Kernel sollte man *zuerst* dieses Dokument lesen. Falls diese und die oben genannte Dokumentation die Fragen nicht klären können, dann kann man sich an die Linux-Newsgruppen wenden. Natürlich sollte man es zuerst mit der Lektüre der Newsgruppe versuchen, anstatt blindlings eine Frage zu schicken. 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.

Allgemeine Fragen über die Systemkonfiguration sollte man an die Newsgruppe

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

richten. Wir *bitten* darum, sich an diese allgemeinen Richtlinien zu halten und Fragen nicht an andere Gruppen zu stellen.

1.4 Neue Versionen dieses Dokuments

Neue Versionen dieses Dokuments erhält man via anonymous FTP unter

```
sunsite.unc.edu:/pub/Linux/docs/HOWTO/
```

Da *SunSITE* normalerweise stark beansprucht ist, ist man besser beraten, sich das Dokument von einer der Linux-FTP-Mirror-Sites zu holen. Neue Versionen erscheinen immer dann, sobald neue Informationen und/oder Treiber verfügbar sind. Sollte diese Kopie hier mehr als ein paar Monate alt sein, sollte man sich möglicherweise informieren, ob nicht bereits eine neue Kopie existiert.

Dieses Dokument wurde mit einem abgewandelten SGML-System geschrieben, welches speziell für das Linux-HOWTO-Projekt geschaffen wurde. Es stehen verschiedene Ausgabe-Formate zur Verfügung, einschließlich Postscript, DVI, ASCII, HTML und bald auch TeXinfo. Ich empfehle die Ansicht im HTML- (mit einem WWW-Browser) oder Postscript/DVI-Format. Beide enthalten Querverweise, die im ASCII-Format verlorengehen.

2 Übersicht über die Bootprompt-Parameter

In diesem Abschnitt wird Software genannt, die zur Weiterleitung von Kernel-Bootparametern zum Kernel selbst verwendet werden kann. Es wird ebenfalls erklärt, wie die Parameter ausgeführt 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. Siehe folgende Beispiele:

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

2.1 LILO (LInux LOader)

LILO (LInux LOader), von Werner Almesberger, ist der am häufigsten verwendete. Er hat die Fähigkeit, verschiedene Kernel zu booten und speichert die Konfigurationsinformation als Klartext-Datei. Die meisten Distributionen werden mit LILO als Default-Boot-Loader ausgeliefert. LILO kann DOS, OS/2, Linux, FreeBSD, etc. ohne Schwierigkeiten booten und ist zudem äußerst flexibel.

Eine typische Konfiguration wird kurz nach dem Systemstart LILO stoppen und folgendes ausgeben: `LILO:`. Der Anwender hat dann einige Sekunden für einen beliebigen Eintrag Zeit. Erfolgt kein Eintrag wird das Default-System gebootet. Typische Systemnamen, die in den LILO-Konfigurationsdateien verwendet werden, sind `linux` und `backup` und `msdos`. Falls ein Bootparameter eingegeben werden soll, wird man dies an dieser Stelle tun, nachdem man den Systemnamen, von dem LILO booten soll, eingegeben hat. Folgendes Beispiel soll dies verdeutlichen:

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

LILO ist hervorragend dokumentiert und für die hier diskutierten Bootparameter ist das LILO-Kommando `append=` von großer Bedeutung, wenn man zur LILO config-Datei einen ständigen Bootparameter hinzufügen will. Man muß einfach nur etwas wie

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

an die `/etc/lilo.conf`-Datei anhängen. Dies kann entweder an die höchste Ebene der config-Datei eingefügt werden und somit für alle Sektoren gelten, oder es soll nur für einen einzigen Systemabschnitt gelten, dann wird es innerhalb eines `image=`-Sektors eingefügt. Eine ausführliche Beschreibung erhält man in der LILO-Dokumentation.

2.2 LoadLin

Ein anderer weitverbreiteter Linux-Loader ist ‘LoadLin’. Dieser ist ein DOS-Programm, das die Fähigkeit besitzt, einen Linux-Kernel vom DOS-Prompt aus zu starten (mit Bootparametern), vorausgesetzt, bestimmte Ressourcen stehen zur Verfügung. Dies ist ein Vorteil für alle, die Linux von DOS aus starten wollen.

Es ist ebenfalls sehr nützlich, wenn man bestimmte Hardware besitzt, welche den zur Verfügung stehenden DOS-Treiber benötigt, um die Hardware in einen bekannten Zustand zu bringen. 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 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.

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 in mehreren Bytes im Kernel-Image selbst gespeichert werden. 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` wird gewöhnlich auch swapdev, ramsize, vidmode und rootflags genannt. Diese Namen zeigen die 5 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[=value_1][,value_2]...[,value_11]
```

`name` ist hierbei ein einzigartiges Schlüsselwort, das Aufschluß darüber gibt, für welchen Teil des Kernels die entsprechenden Werte (falls überhaupt) bestimmt sind. Mehrere Bootparameter werden als Liste nach obigem Format ausgegeben, wobei die einzelnen Parameter durch Leerzeichen getrennt sind. Man beachte das tatsächliche Limit von 11. 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 10 Ganzzahlen- Parameter und eine anschließende Zeichenfolge unterteilt. Das heißt, man kann nicht wirklich 11 Ganzzahlen bereitstellen, höchstens durch Konvertierung des 11ten Parameters von einer Zeichenkette in eine Ganzzahl im Treiber selbst.

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

Der Kernel geht dann durch eine Liste von Setup-Funktionen (zu finden im `bootsetups`-Array), um zu sehen, ob die entsprechende Zeichenkette (wie z.B. `foo`) mit einer Setup-Funktion (`foo_setup()`) für ein bestimmtes Gerät oder einen Teil des Kernels verbunden ist. 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 ja, würde er die Setup-Funktion, die mit ‘foo’ (`foo_setup()`) verbunden ist, aufrufen und dieser die Ganzzahlen-Parameter 3, 4, 5 und 6 übergeben, wie auf der Kernel- Kommandozeile angegeben. Darüberhinaus würde er ebenfalls die Zeichenkette `bar` übergeben.

2.5 Umgebungsvariablen bestimmen.

Alles, was aussieht wie `foo=bar` und nicht als Setup-Funktion akzeptiert wird, wie oben beschrieben, wird dann als festzusetzende Umgebungsvariable interpretiert. Ein (sinnloses?) Beispiel wäre die Verwendung von `TERM=vt100` als Bootparameter.

2.6 Weitergabe von Parametern zum ‘init’-Programm

Alle verbleibenden Parameter, die nicht vom Kernel aufgegriffen und nicht als Umgebungsvariablen interpretiert wurden, werden dann zum ersten Prozeß weitergeleitet. Dieser ist normalerweise das `init`-Programm. Der am häufigsten zum `init`-Prozeß weitergeleitete Parameter ist das Wort *single*, welches `init` anweist, 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 Manpage.

3 Allgemeine, Geräte-unabhängige Bootparameter

Hier handelt es sich um Bootparameter, die mit keinem speziellen Gerät oder Peripheriegerät verknüpft sind. Statt dessen sind sie mit bestimmten internen Kernel-Parametern verbunden, wie z.B. der Umgang mit Speicher, Ramdisk, Root-Dateisystem und andere.

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 Gerät beim Booten als Root-Dateisystem benutzt werden soll. Die Standardeinstellung ist hierbei der Wert des Root-Geräts des Systems, auf dem der Kernel gebaut wurde. Wurde der fragliche Kernel z.B. auf einem System gebaut, das `/dev/hda1` als Root-Partition verwendete, dann wäre das Standard- Root-Gerät `/dev/hda1`. Will man diesen Standard-Wert außer Kraft setzen und das zweite Diskettenlaufwerk als Root-Gerät wählen, würde man `root=/dev/fd1` wählen.

Gültige Root-Geräte:

1. `/dev/hdaN` bis `/dev/hddN`, welches Partition N auf der ST-506-kompatiblen Platte 'a bis d' ist.
2. `/dev/sdaN` bis `/dev/sdeN`, welches Partition N auf der SCSI-kompatiblen Platte 'a bis e' ist.
3. `/dev/xdaN` bis `/dev/xdbN`, welches Partition N auf der XT-kompatiblen Platte 'a bis b' ist.
4. `/dev/fdN`, welches das Floppy-Platten-Laufwerk Nummer N ist. N=0 wäre das DOS-Laufwerk 'A:' und N=1 wäre 'B:'.
5. `/dev/nfs`, was eigentlich kein wirkliches Gerät ist, sondern eher ein Kennzeichen, das den Kernel auffordert, das Root-Dateisystem über das Netzwerk zu holen.

Die schwierigere und weniger portierbare numerische Bestimmung der oben genannten, möglichen Plattengeräte im major/minor-Format wird auch akzeptiert. (z.B. `/dev/sda3` ist major 8, minor 3, d.h. man könnte als Alternative auch `root=0x803` verwenden.)

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 man keine verläßliche Überprüfung der Dateisystem-Integrität vornehmen, während halb-geschriebene Dateien geprüft werden. Mit der Option 'ro' wird der Kernel aufgefordert, das Root-Dateisystem als 'readonly' zu mounten, so daß Programme zur Konsistenzprüfung des Dateisystems (fsck) mit Sicherheit annehmen können, daß keinerlei halb-geschriebene Dateien geprüft werden, während die Überprüfung stattfindet. Kein Programm oder Prozeß kann Dateien des fraglichen Dateisystems beschreiben bis es nicht mit Lese/Schreibrecht wieder gemountet ist.

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

3.1.3 Der Paramater ‘rw’

Dies ist das exakte Gegenteil vom eben genannten. Hier wird der Kernel nämlich aufgefordert, das Root-Dateisystem mit Lese/Schreibrecht zu mounten. Die Default-Einstellung ist sowieso das Mounten des Root-Datei-Systems 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, welches normalerweise für Bootstrapping-Rechner während des Installationsvorgangs verwendet wird, oder für Rechner mit Modultreibern, die zum Zwecke des Zugriffs auf das Root-Dateisystem installiert werden müssen.

3.2.1 Das Kommando ‘ramdisk_start=’

Das Kommando ‘`ramdisk_start=<offset>`’ ermöglicht einem Kernel-Image, zusammen mit einem komprimierten Ramdisk-Image auf einer Floppy zu bleiben. Der Kernel kann nicht in das komprimierte Ramdisk-Dateisystem-Image aufgenommen werden, weil er beginnend mit Block Null gespeichert werden muß, so daß das BIOS den Bootsektor laden kann. Dann kann der Kernel sich selbst erstmalig booten und damit zum Laufen bringen.

Wichtig: Benutzt man ein unkomprimiertes Ramdisk-Image, dann kann der Kernel Teil des Dateisystem-Images sein, das in die Ramdisk geladen wird und die Floppy kann mit LILO gebootet werden. Die beiden können auch getrennt sein wie bei den komprimierten Images.

Verwendet man ein 2-Disketten Boot/Root-Setup (Kernel auf Diskette 1, Ramdisk-Image auf Diskette 2) dann startet die Ramdisk 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 er ein Ramdisk-Image laden soll oder nicht. ‘`load_ramdisk=1`’ soll den Kernel veranlassen, eine Floppy in die Ramdisk zu laden. Der Standardwert ist Null, was bedeutet, daß der Kernel nicht versuchen soll, eine Ramdisk 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 bestimmt und via `rdev` im Kernel-Image gespeichert werden kann.

3.2.3 Das Kommando ‘prompt_ramdisk=’

Dieser Parameter teilt dem Kernel mit, ob er einen Prompt ausgeben soll mit dem man aufgefordert wird, die Floppy mit dem Ramdisk-Image einzulegen. Bei einer 1-Disketten-Konfiguration befindet sich das Ramdisk-Image auf der gleichen Floppy wie der Kernel, der gerade den Lade-/Bootvorgang beendet hat. Somit wird ein Prompt nicht benötigt. In diesem Fall kann man das Kommando ‘`prompt_ramdisk=0`’ verwenden. Bei einer 2-Disketten- Konfiguration 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. (Geschichtliche Anmerkung: Heimtückische Anwender verwendeten gewöhnlich die Option ‘`vga=ask`’ LILO, um den Bootprozeß zeitweilig zu stoppen und ein Umschalten von der Boot- zur Rootfloppy 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 Ramdisk je nach Bedarf dynamisch wächst, jedoch gibt es eine Obergrenze für ihre Größe, so daß nicht der gesamte RAM verbraucht wird und einem Schwierigkeiten erspart werden. Der Standardwert ist 4096 (4MB), 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)

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

Dies gibt in KB die Größe des RAM-Disk-Devices an. Würde man z.B. ein Root-Dateisystem auf einer 1,44 MB Floppy 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 somit mit dem Hilfsprogramm `rdev` verändert werden kann.

3.2.6 Das Kommando 'noinitrd' (initial RAM-Disk)

Kernel v2.x und neuere verfügen über ein Feature, bei dem das Root- Dateisystem anfangs eine RAM-Disk ist und der Kernel `/linuxrc` auf diesem RAM-Image ausführt. Dieses Feature wird typischerweise dazu verwendet, das Laden von Modulen zu ermöglichen, die zum Mounten des eigentlichen Root-Dateisystems benötigt werden (z.B. Laden der SCSI-Treiber-Module, die im RAM-Disk-Image gespeichert sind und anschließendes Mounten des eigentlichen Root-Dateisystems auf einer SCSI-Diskette.)

Der eigentliche 'noinitrd'-Parameter bestimmt, was mit den initrd-Daten passiert, nachdem der Kernel gebootet hat. Wenn Sie bestimmt werden, statt auf eine RAM-Disk konvertiert zu werden, sind die Daten erhältlich unter `/dev/initrd`, welche eingesehen werden kann bevor der RAM wieder dem System übergeben 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 nach, wie Linux den physikalischen oder virtuellen Systemspeicher erkennt oder behandelt.

3.3.1 Der Parameter 'mem='

Dieser Parameter dient zwei verschiedenen Zwecken: Der ursprüngliche Zweck war die Bestimmung der installierten Speicherkapazität (oder ein kleinerer Wert, falls man den Speicherplatz für Linux verringern wollte). Der andere (und

kaum verwendete) Zweck ist die Bestimmung von `mem=nopentium`, das den Linux-Kernel auffordert, das 4 MB Seitentabellen-Performance-Feature nicht zu verwenden.

Der ursprüngliche BIOS-Aufruf, definiert in der PC-Spezifikation, der den installierten Speicherplatz wiedergibt, wurde nur deswegen eingeführt, um bis zu 64 MB angeben zu können. (Richtig, wieder mangelnde Voraussicht, genau wie bei den 1024 Zylinder-Platten... seufz.) Linux verwendet diesen BIOS-Aufruf beim Booten zur Bestimmung des installierten Speicherplatzes. 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:

> The kernel will accept any '`mem=xx`' parameter you give it, and if it turns out that you lied to it, it will crash horribly sooner or later. The parameter indicates the highest addressable RAM address, so '`mem=0x1000000`' means you have 16 MB of memory, for example. For a 96 MB machine this would be '`mem=0x6000000`'.

WICHTIG WICHTIG WICHTIG: Einige Rechner verwenden möglicherweise den größtmöglichen Speicherplatz für das Bereithalten des BIOS im Cache o. ä., so daß man möglicherweise nicht wirklich die vollen 96 MB belegen kann. Auch das Gegenteil trifft zu: Einige Chipsets werden den physikalischen Speicher, der vom BIOS belegt wird, auf den Bereich über dem Speichermaximum abbilden, d.h. der maximale Speicher wird in Wirklichkeit z.B. 96 MB + 384 kB betragen. 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 Parameter keine Hexadezimalzahl sein muß und daß die Endungen '`k`' und '`M`' (egal, ob Groß- oder Kleinschreibung) zur Bestimmung von Kilobytes, beziehungsweise Megabytes verwendet werden können. ('`k`' verursacht auf dem Wert eine Verschiebung von 10 Bit und '`M`' verursacht eine Verschiebung um 20 Bit.) Die oben genannte Warnung hat insofern noch Gültigkeit, daß ein 96 MB Rechner mit `mem=97920k` laufen mag, jedoch mit `mem=98304k` oder `mem=96M` versagt.

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 8 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 auf die 'Goodies' in `/proc/sys/vm` werfen.

3.3.3 Der Parameter 'buff='

Ähnlich dem 'swap='-Parameter wird dem Anwender die Feineinstellung einiger der Parameter für die Buffer-Speicherverwaltung ermöglicht. Folgende 6 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 auf die 'Goodies' in `/proc/sys/vm` werfen.

3.4 Bootparameter für das NFS-Root-Dateisystem

Linux unterstützt Systeme wie laufwerkslose Workstations, indem ihr Rootfilesystem als NFS (Network FileSystem) besteht. 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 Information lediglich aus einer Zusammenfassung dieser Datei besteht.

3.4.1 Der Parameter 'nfsroot='

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

```
nfsroot=[<server-ip>:]<root-dir>[,<nfs-options>]
```

Wird der nfsroot-Parameter nicht auf der Kommandozeile angegeben, dann wird defaultmäßig '`/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-dir>

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-options>

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 übers Netzwerk benötigt werden. Wird dieser Parameter nicht angegeben, dann versucht der Kernel die Verwendung von RARP und/oder BOOTP, um diese Parameter herauszufinden. Dies sieht folgendermaßen aus:

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

<my-ip>

IP-Adresse des Clients. Ist dieses Feld leer, wird die Adresse entweder von RARP oder von BOOTP bestimmt. Welches Protokoll verwendet wird, hängt vom <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 im nfsroot- Parameter fest (siehe oben). Bleibt dieser Eintrag aus, dann wird die Adresse des Servers verwendet, welcher auf die RARP- oder BOOTP-Anfrage antwortete.

<gw-ip>

IP-Adresse eines Gateways, falls der Server sich auf einem anderen Subnetz befindet. Ist dieser Eintrag leer, dann wird kein Gateway verwendet und es wird angenommen, daß sich der Server auf dem lokalen Netzwerk befindet, bis BOOTP einen Wert erhält.

<netmask>

Netzmaske für die lokale Netzwerkschnittstelle. Ist dieser Eintrag leer, dann wird die Netzmaske von der Client-IP-Adresse abgeleitet, bis BOOTP einen Wert erhält.

<name>

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

<dev>

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

<auto>

Zu verwendende Methode für die AutoKonfiguration. Ist dies 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 AutoKonfiguration 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 (ohne die ganzen ':' Zeichen davor). In diesem Fall wird Autokonfiguration 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 Operator. Wird die Nachricht als wichtig angesehen, dann wird die Funktion `printk()` eine Kopie auf die aktuelle Konsole senden und sie an das Hilfsprogramm `klogd()` aushändigen, so daß sie auf Festplatte aufgezeichnet wird. Der Grund für die Ausgabe wichtiger Nachrichten auf der Konsole und gleichzeitiger Protokollierung durch die Festplatte liegt

darin, daß unter unglücklichen Umständen (z.B. ein Plattenausfall) die Nachricht nicht zur Festplatte gelangt und somit verlorengeht.

Der Level 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 festgehalten. (Die verschiedenen Level sind in der include-Datei `kernel.h`) definiert. Das Festlegen von `debug` als Bootparameter setzt den Loglevel der Konsole auf 10, so daß *alle* Kernel- Mitteilungen auf der Konsole erscheinen.

Der Loglevel der Konsole kann normalerweise auch bei der Ausführung mittels einer Option an das Programm `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` (herabgesetzt) 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 den 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 'no387'

Einige i387 Koprozessor-Chips haben Bugs, die sich bei der Verwendung im 32 bit-geschützten Modus zeigen. Einige der frühen ULSI-387 Chips verursachen z.B. massive Totalsperren während der Ausführung von Fließpunkt-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 mit mathematischer Emulations-Unterstützung kompiliert sein! Dies ist möglicherweise auch dann sinnvoll, wenn man einen dieser *wirklich* alten 386er hat, die einen 80287 FPU verwenden können, da Linux keinen 80287 verwenden kann.

3.5.4 Der Parameter 'no-hlt'

Die i386-Familie der 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. Dies erlaubt der CPU in einen 'low-power' Modus einzutreten, wo sie wie ein Zombie verharrt, bis sie von einem externen Gerät geweckt wird (gewöhnlich durch einen Interrupt). Einige der frühen i486DX-100 Chips 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 laufen zu lassen, wenn es nichts anderes zu tun gibt und die CPU *nicht* zu stoppen, wenn es keine aktiven Prozesse gibt. Dies ermöglicht Benutzern mit solchen kaputten Chips die Verwendung von Linux, obwohl sie gut beraten wären, sich durch eine möglicherweise vorhandene Garantie einen Ersatz zu suchen.

3.5.5 Der Parameter 'no-scroll'

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

3.5.6 Der Parameter 'panic='

Für den unwahrscheinlichen Fall einer Kernel-Panic (ein interner Fehler, der vom Kernel erkannt wurde und den der Kernel als ernst genug erachtet, um sich laut zu beschweren und dann alles zu stoppen), wird für gewöhnlich gewartet,

bis jemand vorbeikommt, die Panik-Message auf dem Bildschirm entdeckt und den Rechner neu bootet. 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-Panic versuchen würde, sich selbst zu booten. Standardwert ist Null und führt zum Standardverhalten, das darin besteht, endlos zu warten.

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

3.5.7 Der Parameter 'profile='

Kernel-Entwickler können eine Option aktivieren, die es ihnen erlaubt, festzulegen, 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 automatischer Aktivierung des Profiling kompilieren. In jedem Fall braucht man ein Tool wie `readprofile.c` , das die Ausgabe `/proc/profile` verwenden kann.

3.5.8 Der Parameter 'reboot='

Diese Option kontrolliert die Art des Reboots, die Linux beim Reset des Rechners ausführt (normalerweise via `/sbin/init` das Control-Alt-Delete ausführt). Seit den späten v2.0-Kerneln 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 dies bei billiger/kaputter Hardware, die es nicht schafft neu zu booten, wenn ein Warmstart erforderlich wäre, normalerweise funktioniert. Zum Wiederherstellen des alten Zustands (hier: Warmstart) 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 wahrnehmen und Daten vom Cache auf die Festplatte schreiben. Nach einem Kaltstart wird die Speicherkarte möglicherweise zurückgeschaltet und die write-back-Daten im Cache-Speicher gehen verloren. Andere haben Systeme, die lange für den Speicher-Check brauchen und/oder SCSI BIOSe, die nach einem Kaltstart länger für die Initialisierung brauchen als guten Grund für den Einsatz eines Warmstarts gemeldet.

3.5.9 Der Parameter 'reserve='

Dieser wird dazu benutzt, I/O Port-Bereiche vor Überprüfungen zu *schützen*. Das Kommando ist folgendermaßen aufgebaut:

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

Bei einigen Rechnern mag es notwendig sein, Gerätetreiber davon abzuhalten, in einer bestimmten Region nach Geräten zu suchen (automatische Hardwareerkennung). 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 Port-Registrationstabelle des Kernels so behandelt, als ob dort bereits ein Gerät gefunden wurde (trägt den Namen `reserved`). 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 sind gegen eine Geräteerkennung geschützt, bei der vorrangig `check_region()` ausgeführt wird, statt daß blind ein I/O Bereich geprüft wird. Dies wird dann eingesetzt, wenn Treiber auf einem NE2000 hängenbleiben oder irrtümlich andere Geräte als eigene erkannt werden. Ein korrekter Gerätetreiber sollte keine reservierten

Bereiche prüfen, wenn nicht ein anderer Bootparameter dies 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 eine genaue Erkennung für dieses Gerät bestimmen. Die meisten Treiber ignorieren die Port- Registrations-Tabelle, 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 11 Parametern, d.h. man kann nur 5 reservierte Bereiche pro `reserve` Schlüsselwort bestimmen. Bei ungewöhnlich komplizierten Anfragen sind jedoch mehrere `reserve` Argumente möglich.

3.5.10 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 Erwähnug verdient. Sie kann auch durch die Verwendung von `rdev -v` festgelegt werden und ebenso durch `vidmode` auf der Datei vmlinuz. Dies 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 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.

4.1 Parameter für MIDLEVEL-Treiber

MIDLEVEL-Treiber sind zuständig für Dinge wie Disketten, CD-ROMs und Tapes ohne dabei in Host-Adapter-Bezeichnungen zu geraten.

4.1.1 Maximale Anzahl überprüfter LUNs ('max_scsi_luns=')

Jedes SCSI-Gerät kann eine Reihe von 'Sub-Geräten' enthalten. Das beste Beispiel ist eines der neuen SCSI CD-ROMs, das gleichzeitig mehr als eine CD bedienen kann. Jede CD 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.

Das Problem ergibt sich bei Geräten mit einer einzigen LUN mit schlechter Firmware. Einige schlecht entwickelte SCSI-Geräte (alte und unglücklicherweise auch neue) können nicht mit der Überprüfung für LUNs ungleich Null umgehen. Sie antworten, indem sie sich und möglicherweise mit sich den gesamten SCSI-Bus sperren.

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.

4.1.2 Parameter für den SCSI-Tape-Treiber ('st=')

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

```
st=buf_size[,write_threshold[,max_bufs]]
```

Die ersten zwei Zahlen werden in kB-Einheiten angegeben. Die Standard- `buf_size` ist 32 kB und die maximal zu bestimmende Größe sind lächerliche 16384 kB. `write_threshold` ist der Wert bei dem der Puffer an das Tape 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.

4.2 Parameter für SCSI-Host-Adapter

Allgemeine Anmerkungen zu diesem Abschnitt:

`iobase`

Der erste I/O Port, der vom SCSI-Host besetzt wird. Diese werden in der Hexadezimalschreibweise angegeben und liegen 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 (Direct Memory Access)- Kanal. Dies gilt typischerweise nur für Bus-Kontroller-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 die meisten ihn ständig intern bestimmen lassen. Normal ist der Standardwert 7. Die Seagate und Future Domain TMC-950-Boards jedoch verwenden 6.

`parity`

Ob der SCSI Host-Adapter von den angeschlossenen Geräten erwartet, daß sie einen Paritätswert mit dem gesamten Informationsaustausch zur Verfügung stellen. Der Wert 1 aktiviert den Paritäts-Check und Null schaltet ihn ab. Auch hier gilt, daß nicht alle Adapter die Auswahl eines Paritätsverhaltens als Bootparameter unterstützen.

4.2.1 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.

Der Prüf-Code für diese SCSI Hosts schaut nach einem installierten BIOS. Falls keines vorhanden ist, dann wird die Karte nicht gefunden. Für diesen Fall wird man einen Bootparameter folgender Form verwenden müssen:

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

Beachte: Wurde der Treiber bei aktiviertem Debugging kompiliert, dann kann ein sechster Wert zur Bestimmung des Debug-Levels bestimmt werden.

Alle Parameter verhalten sich wie eingangs dieses Abschnitts beschrieben und der `reconnect` Wert erlaubt die Abstöpselung oder ein erneutes Anschließen des Geräts, falls ein Wert ungleich Null benutzt wird. 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 Umklemmwert angeben.

4.2.2 Adaptec aha154x (‘aha1542=’)

Dies sind die Karten aus der aha154x-Serie. Die Karten aus der aha1542-Serie verfügen über einen i82077 Floppy-Kontroller, die aha1540 Serienkarten 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, 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. Standardmäßig gilt: 11 us an und 4 us 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 Geschwindigkeitsrate (in MB/s) in der der DMA (Direct Memory Access)-Transfer erfolgt. Defaultwert ist 5 MB/s. Karten einer neueren Revision ermöglichen die Auswahl dieses Werts als Teil der Soft-Konfiguration, 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.

4.2.3 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, den SCSI-Bus nicht zurückzuschalten, wenn der Host-Adapter beim Booten bestimmt wird.

4.2.4 AdvanSys SCSI Host-Adapter (‘advansys=’)

Der AdvanSys-Treiber akzeptiert bis zu 4 I/O Adressen, die für eine AdvanSys SCSI-Karte überprüft werden. Man beachte, daß diese Werte (falls verwendet) überhaupt keinen Einfluß auf die EISA- oder PCI- Überprüfung haben. Sie werden nur für die Überprüfung von ISA- und VLB-Karten eingesetzt. Wurde der Treiber bei eingeschalteter Fehlererkennung kompiliert, kann zusätzlich der Level der Fehlermeldung durch Hinzufügen des Parameters `0xdeb`

$$0 - f$$

bestimmt werden. `0-f` ermöglicht die Festlegung des Levels der Fehlermeldungen auf jeden der 16 Wort-Level.

4.2.5 Always IN2000-Host-Adapter (‘in2000=’)

Im Gegensatz zu anderen SCSI Host-Bootparametern verwendet der IN2000-Treiber für die meisten seiner Ganzzahlen-Parameter ASCII-String-Vorsilben. Hier eine Liste von unterstützten Parametern:

ioport:addr

Wobei `addr` die IO-Adresse einer (gewöhnlich ROM-losen) Karte ist.

noreset

Keine optionalen Parameter. Verhindert die Zurückschaltung des SCSI-Bus beim Booten.

nosync:x

`x` ist eine Bitmaske, wobei die ersten 7 Bit mit den 7 möglichen SCSI-Geräten übereinstimmen (Bit 0 für Gerät #0, etc). Um sync-Übertragung auf diesem Gerät zu VERHINDERN, bestimmt man ein Bit. Treiber-Standard ist sync auf allen Geräten DEAKTIVIERT.

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 das Abschalten, 2 erlaubt das Abschalten. `x = 1` führt ’adaptive’ Unterbrechungen durch. Dies ist Standard, und wohl allgemein die beste Wahl.

debug:x

Ist ‘DEBUGGING_ON’ angegeben, dann ist `x` eine Bitmaske, durch die verschiedene Debug-Ausgaben gedruckt werden - see the DB_xxx defines 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 - see the PR_xxx defines in `in2000.h`.

Hier einige Anwendungs-Beispiele :

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

4.2.6 AMD AM53C974-basierte Hardware (‘AM53C974=’)

Im Gegensatz zu anderen Treibern verwendet dieser keine Bootparameter, um I/O-, IRQ- or DMA-Kanäle mitzuteilen. (Da AM53C974 ein PCI-Gerät ist, sollte dafür auch kein Grund bestehen.) Stattdessen 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 Kontroller mit SCSI-ID 7 und dem Gerät mit SCSI-ID 2 sollte man eine Transferrate von 8 MHz im Synchronmodus mit max. 15 Bytes Abweichung in Betracht ziehen.’ Weitere Informationen findet man in der Datei `linux/drivers/scsi/README.AM53C974`

4.2.7 BusLogic SCSI-Hosts mit v1.2 Kerneln (‘buslogic=’)

In älteren Kerneln akzeptiert der buslogic-Treiber nur einen Parameter, und zwar die I/O base. Sie erwartet, einer der folgenden Werte zu sein: `0x130, 0x134, 0x230, 0x234, 0x330, 0x334`.

4.2.8 BusLogic SCSI Hosts mit v2.x Kerneln (‘BusLogic=’)

Bei v2.x Kerneln akzeptiert der BusLogic-Treiber viele Parameter. (Man beachte obige Schreibweise; Groß B und L!!!). Die folgende detaillierte Beschreibung ist direkt Leonard N. Zubkoffs Treiber des v2.0 Kernels entnommen.

Beim BusLogic-Treiber umfaßt der Kernel-Kommandozeilen-Eintrag den Treiber-Identifier ”BusLogic=” , optional gefolgt von einer durch Komma getrennten Ganzzahlenfolge und darauf optional gefolgt von einer durch Komma getrennten Stringfolge. Jeder Kommandozeilen- Eintrag gilt für einen BusLogic Host-Adapter. Bei Systemen, die mehrere BusLogic Host-Adapter enthalten, können mehrere Kommandozeilen-Einträge verwendet werden.

Die erste angegebene Ganzzahl ist die I/O Addresse, auf der sich der Host-Adapter befindet. Fehlt diese Angabe wird der Standardwert 0 benutzt. Das bedeutet, dieser Eintrag gilt für den ersten BusLogic-Host-Adapter, der während der Standard- Überprüfungs-Sequenz gefunden wurde. Wurden irgendwelche I/O Adressen-Parameter auf der Kommandozeile angegeben, dann wird die Standard-Überprüfungs-Sequenz ausgelassen.

Die zweite angegebene Ganzzahl ist die Tagged Queue Depth, die für Zielgeräte, die Tagged Queuing unterstützen, zu verwenden ist. Die Queue Depth ist die Anzahl von SCSI-Kommandos, die gleichzeitig als auszuführend angegeben werden dürfen. Fehlt eine Angabe wird standardmäßig 0 verwendet, d.h. es wird ein automatisch bestimmter Wert verwendet, basierend auf der maximalen Queue Depth des Host-Adapters und der Anzahl, dem Typ, der Geschwindigkeit und den Fähigkeiten der erkannten Zielgeräte. Bei Host-Adaptern, die ISA Bounce Buffers benötigen, wird die Tagged Queue Depth automatisch auf BusLogic_TaggedQueueDepth_BB gesetzt, um ausschweifende Vor-Zuordnung von DMA Bounce Buffer-Speicher zu vermeiden. Zielgeräte, die Tagged Queuing nicht unterstützen verwenden eine Queue Depth von BusLogic_UntaggedQueueDepth.

Die dritte angegebene Ganzzahl ist die Bus Settle Time in Sekunden. Dies gibt die Zeit an, die man zwischen einem Host Adapter Hard Reset, der einen SCSI Bus Reset einleitet, und der Eingabe von SCSI-Kommandos wartet. Fehlt eine Angabe, dann wird standardmäßig 0 verwendet, d.h. es wird der Wert von BusLogic_DefaultBusSettleTime verwendet.

Die vierte angegebene Ganzzahl sind die Local Options. Fehlt die Angabe, dann wird standardmäßig 0 verwendet. Man beachte, daß Local Options nur für einen bestimmten Host-Adapter gelten.

Die fünfte angegebene Ganzzahl sind die Global Options. Fehlt die Angabe wird standardmäßig 0 verwendet. Man beachte, daß Global Options auf alle Host Adapter angewendet wird.

Die String-Optionen dienen der Kontrolle über Tagged Queuing, Error Recovery (Fehlerkorrektur) und Host-Adapter-Überprüfung.

Die Tagged Queuing-Bestimmung beginnt mit "`TQ:`" und erlaubt die ausdrückliche Festlegung, ob Tagged Queuing auf Zielgeräten, die dies unterstützen, erlaubt ist. Folgende Optionen zur Bestimmung stehen zur Verfügung:

TQ:Default

Ob Tagged Queuing erlaubt ist, hängt von der Firmware- Version des BusLogic Host-Adapters ab und davon, ob es der Tagged Queue Depth-Wert erlaubt, mehrere Kommandos in die Warteschlange zu stellen.

TQ:Enable

Tagged Queuing wird für alle Zielgeräte auf diesem Host-Adapter aktiviert, wobei jegliche Beschränkungen übergangen werden, die ansonsten, basierend auf der Host Adapter Firmware-Version, auferlegt werden würden.

TQ:Disable

Tagged Queuing wird für alle Zielgeräte auf diesem Host-Adapter deaktiviert.

TQ:<Per-Target-Spec>

Tagged Queuing wird für jedes einzelne Zielgerät kontrolliert. <Per-Target-Spec> ist eine Sequenz aus "Y", "N", und "X" Zeichen. "Y" aktiviert Tagged Queuing, "N" deaktiviert Tagged Queuing und "X" akzeptiert den Standardwert, basierend auf der Firmware-Version. Das erste Zeichen bezieht sich auf Zielgerät 0, das zweite auf Zielgerät 1 usw.; falls die Sequenz aus "Y", "N" und "X" Zeichen nicht alle Zielgeräte abdeckt, werden nicht spezifizierte Zeichen als "X" angenommen.

Man beachte, daß das ausdrückliche Verlangen nach Tagged Queuing zu Problemen führen kann; diese Möglichkeit dient primär dazu, das Deaktivieren von Tagged Queuing auf Zielgeräten zu ermöglichen, die es nicht korrekt durchführen.

Die Error Recovery Strategy-Spezifikation beginnt mit "`ER:`" Sie ermöglicht, ausdrücklich festzulegen, daß die Error Recovery- Aktion ausgeführt wird, wenn das ResetCommand aufgerufen wird - aufgrund eines Versagens des SCSI-Kommandos, erfolgreich abzulaufen. Folgende Optionen zur Bestimmung stehen zur Verfügung:

ER:Default

Ausgehend von der Empfehlung des SCSI-Subsystems wird Error Recovery zwischen einem Hard Reset und den Bus Device Reset-Optionen wählen.

ER:HardReset

Error Recovery wird einen Host Adapter Hard Reset einleiten, was zusätzlich einen SCSI Bus Reset verursacht.

ER:BusDeviceReset

Error Recovery wird eine Bus Device Reset-Mitteilung zu dem jeweiligen Zielgerät schicken, das den Fehler verursacht hat. Wird noch einmal Error Recovery für dieses Zielgerät eingeleitet, und hat kein SCSI-Kommando an dieses Zielgerät erfolgreich ausgeführt, seit die Bus Device Reset-Nachricht geschickt wurde, dann wird ein Hard Reset nötig sein.

ER:None

Error Recovery wird unterdrückt. Diese Option sollte nur dann gewählt werden, wenn ein SCSI Bus-Reset oder Bus Device-Reset das Zielgerät veranlaßt, komplett und unwiderruflich zu versagen.

ER:<Per-Target-Spec>

Error Recovery wird für jedes einzelne Zielgerät kontrolliert. <Per-Target-Spec> ist eine Sequenz aus "D", "H", "B" und "N" Zeichen. Mit "D" wird Default gewählt, "H" steht für Hard Reset, "B" für Bus Device-Reset und mit "N" wird None gewählt. Das erste Zeichen bezieht sich auf Zielgerät 0, das zweite auf Zielgerät 1 usw. werden mit "D", "H", "B" und "N" nicht alle möglichen Zielgeräte abgedeckt, dann werden undefinierte Zeichen als "D" angenommen.

Die Host-Adapter-Überprüfungs-Spezifikation umfaßt folgende Zeichenketten:

NoProbe

Es soll keine Überprüfung stattfinden; somit werden keine BusLogic-Host-Adapter erkannt.

NoProbeISA

Die Standard ISA I/O Adresssen werden nicht untersucht; somit werden nur PCI-Host-Adapter erkannt.

NoSortPCI

PCI-Host-Adapter werden in der vom PCI BIOS bereitgestellten Reihenfolge aufgezählt, wobei die Einstellungen der Option AutoSCSI ”Use Bus And Device # For PCI Scanning Seq.” ignoriert werden.

4.2.9 EATA SCSI-Karten (‘eata=’)

Seit den späten Kernelversionen v2.0 akzeptieren die EATA-Treiber die Überprüfung eines Bootparameters, der die i/o base(s) bestimmt. Dies sieht folgendermaßen aus:

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

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

4.2.10 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. Oder, wenn die Signature-String des BIOS nicht erkannt wurde, dann wird die Karte auch 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 speicherkonformen I/O-Bereichs, den die Karte verwendet. Dieser ist für gewöhnlich einer der folgenden Werte: `0xc8000, 0xca000, 0xcc000, 0xce000, 0xdc000, 0xde000`.

4.2.11 Future Domain TMC-16xx, TMC-3260, AHA-2920 (‘fdomain=’)

Der Treiber erkennt diese Karten entsprechend einer Liste von bekannten BIOS ROM-Signatures. 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]
```

4.2.12 IOMEGA Parallel Port / ZIP-Laufwerk (‘ppa=’)

Dies ist ein Treiber für den IOMEGA Parallel Port SCSI Adapter, welcher in den IOMEGA ZIP-Laufwerken enthalten ist. Es 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 festgelegte Werte. Will man einen der unverbindlichen Parameter verändern, dann sollte man einen Blick auf die Datei `linux/drivers/scsi/README.ppa` werfen. Dort findet man genaue Informationen darüber, was von diesen Parametern kontrolliert wird.

4.2.13 NCR5380-basierte Kontroller (‘ncr5380=’)

Je nach Board kann der 5380 entweder I/O- oder speicherkonform sein. (Eine Adresse unter 0x400 ist für gewöhnlich I/O-konform. PCI- und EISA-Hardware verwenden jedoch i/o-Adressen über 0x3ff.) In jedem Fall gibt man die Adresse, den IRQ-Wert und den DMA-Channel-Wert an. Hier ein Beispiel einer I/O-konformen 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 automatische Hardwareerkennung. Weitere Informationen findet man in der Datei `linux/drivers/scsi/README.g_NCR5380`

4.2.14 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-Channel verwendet wird.

4.2.15 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 Null 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. Der Treiber kann eine Option für die Kompilierungszeit nehmen.

4.2.16 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 Interrupts, jedoch mit einem Leistungsverlust, zu funktionieren. Die `iobase` ist gewöhnlich `0x388`.

4.2.17 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. Oder, wenn die Signature-Zeichenkette des BIOS nicht erkannt wird, dann wird sie auch nicht gefunden. In jedem Fall wird man einen Bootparameter folgender Form verwenden:

```
st0x=mem_base,irq
```

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

4.2.18 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, 0xd8000`.

4.2.19 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.

4.2.20 Western Digital WD7000-Karten (‘wd7000=’)

Der Treiber-Check für wd7000 sucht nach einer bekannten BIOS ROM- Zeichenkette und kennt einige Standard-Konfigurations-Einstellungen. 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
```

4.3 SCSI-Host-Adapter, die keine Bootparameter akzeptieren

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

```
Adaptec aha1740 (EISA-\"{U}berpr\"{u}fung),
NCR53c7xx,8xx (PCI, beide Treiber)
Qlogic Fast (0x230, 0x330)
Qligic ISP (PCI)
```

5 Festplatten

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

5.1 IDE Disk/CD-ROM-Treiber-Parameter

Der IDE-Treiber akzeptiert eine Reihe von Parametern, die von Plattengeometrie-Spezifikationen 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 auf die Datei `ide.txt` im Verzeichnis `linux/Documentation` werfen, dem diese Zusammenfassung entnommen wurde.

```
"hdx="  wird f\"{u}r alle "x" von "a" bis "h", wie z.B. "hdc" erkannt.
"idex=" wird f\"{u}r alle "x" von "0" bis "3", wie z.B. "ide1" erkannt.

"hdx=noprobe"          : Treiber mag vorhanden sein, jedoch soll keine
                         \"{U}berpr\"{u}fung stattfinden
"hdx=none"             : Treiber ist NICHT vorhanden, cmos soll
                         ignoriert werden
                         und eine \"{U}berpr\"{u}fung soll nicht
                         stattfinden
"hdx=nowerr"           : Das WRERR_STAT-Bit auf diesem Laufwerk soll
                         ignoriert werden
"hdx=cdrom"            : Laufwerk ist vorhanden und
                          ist ein CDROM-Laufwerk
"hdx=cyl,head,sect"    : Plattenlaufwerk ist vorhanden, mit angegebener
                         Geometrie
"hdx=autotune"         : der Treiber wird versuchen, die Schnittstellen-
                         Geschwindigkeit auf den schnellsten
                         unterst\"{u}tzten PIO-Modus einzustellen; falls
                         m\"{o}glich, nur f\"{u}r dieses Laufwerk.
                         Wird nicht von allen Chipset-Typen voll
                         unterst\"{u}tzt
                               Verursacht wahrscheinlich Probleme bei
                               \"{a}lteren/odd IDE-Laufwerken.

"idex=noprobe"         : Es soll nicht versucht werden, auf diese
                         Schnittstelle zuzugreifen oder sie zu verwenden
"idex=base"            : F\"{u}hre Schnittstellen-Check bei der angegebenen
                         Adresse aus,
                               wobei "base" normalerweise 0x1f0 oder 0x170 ist
                               und angenommen wird, da{\ss} "ctl"  "base"+0x206 ist.
"idex=base,ctl"        : bestimme sowohl base und ctl
"idex=base,ctl,irq"    : bestimme base, ctl, und irq-Nummer
"idex=autotune"        : der Treiber wird versuchen, die Schnittstellen-
                         Geschwindigkeit auf den schnellsten
                         unterst\"{u}tzten PIO-Modus einzustellen,
                         und zwar f\"{u}r alle auf dieser Schnittstelle
                         befindlichen Laufwerke.
                               Wird nicht von allen Chipset-Typen voll
                               unterst\"{u}tzt
                               Verursacht wahrscheinlich Probleme bei
                               \"{a}lteren/odd IDE-Laufwerken.
"idex=noautotune"      : Treiber versucht NICHT, die Schnittstellen-
                         Geschwindigkeit einzustellen.
                               Dies ist der Standard f\"{u}r die meisten
                               Chipsets,
                               mit Ausnahme von cmd640.
"idex=serialize"       : keine \"{u}berlappenden Aktionen auf idex und
                         ide(xˆ1)
```

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

```
"ide0=dtc2278"         : pr\"{u}fe/unterst\"{u}tze DTC2278 Schnittstelle
"ide0=ht6560b"         : pr\"{u}fe/unterst\"{u}tze HT6560B Schnittstelle
"ide0=cmd640_vlb"      : *NOTWENDIG* f\"{u}r VLB-Karten mit dem CMD640 Chip
                         (nicht f\"{u}r PCI -- wird automatisch erkannt)
```

```
"ide0=qd6580"          : pr\"{u}fe/unterst\"{u}tze qd6580 Schnittstelle
"ide0=ali14xx"         : pr\"{u}fe/unterst\"{u}tze ali14xx Chipsets
                         (ALI M1439/M1445)
"ide0=umc8672"         : pr\"{u}fe/unterst\"{u}tze umc8672 Chipsets
```

Alles übrige wird mit der Nachricht ”BAD OPTION” abgewiesen.

5.2 Standard ST-506-Platten-Treiber-Optionen (‘hd=’)

Der Standard-Plattentreiber kann, ähnlich wie der IDE-Treiber, Geometrie-Parameter für die Platten akzeptieren. Man beachte jedoch, daß er nur drei Werte (C/H/S) erwartet – nur einer mehr oder weniger, und man 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.

5.3 XT-Platten-Treiber-Optionen (‘xd=’)

Sollten Sie zu den Unglücklichen gehören, die eine dieser alten 8-Bit-Karten verwenden, die Daten keuchend mit 125kB/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. Oder, falls der Signature-String Ihres BIOS nicht erkannt wurde, dann 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 Beispiel-Verwendung für einen WD1002- Kontroller mit ausgeschaltetem/entferntem BIOS, wobei die ‘Default’ Parameter vom XT-Kontroller verwendet werden:

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

6 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 die meisten dieser CD-ROMs Dokumentationsdateien enthalten, die man lesen *sollte* . Sie alle sind günstig plaziert: `linux/Documentation/cdrom`.

6.1 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 sowieso laufen. Alle übrigen Werte werden ignoriert.

6.2 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 Hardware-Interrupts nicht unterstützt werden (wie auf einigen PAS-Karten). Unterstützt die eigene Karte Interrupts, dann sollte man diese nutzen, da diese die CPU-Last des Treibers verringern.

Die ‘is_pas_card’ sollte als ‘PAS’ eingetragen werden, falls eine Pro Audio Spectrum-Karte verwendet wird. Andernfalls sollte sie überhaupt nicht bestimmt werden.

6.3 Die CDU-535 Sony-Schnittstelle (‘sonycd535=’)

Diese CD-ROM-Schnittstelle hat folgende Syntax:

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

Für die I/O base kann eine Null als ‘Platzhalter’ verwendet werden, falls man einen IRQ-Wert bestimmen will.

6.4 Die GoldStar-Schnittstelle (‘gscd=’)

Diese CD-ROM-Schnittstelle hat folgende Syntax:

```
gscd=iobase
```

6.5 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.

6.6 Die Mitsumi Standard-Schnittstelle (‘mcd=’)

Die 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, und wird je nach einem Kompilierungszeit-`DEFINE` implementiert oder nicht.

6.7 Die Mitsumi XA/MultiSession-Schnittstelle (‘mcdx=’)

Zur Zeit verfügt dieser ‘experimental’-Treiber über eine Setup- Funktion, jedoch sind bis jetzt (seit 1.3.15) keine Parameter implementiert. Dies gilt für dieselbe Hardware wie oben genannt, aber der Treiber verfügt über erweiterte Features.

6.8 Die Optics Storage-Schnittstelle (‘optcd=’)

Dieser Kartentyp hat folgende Syntax:

```
optcd=iobase
```

6.9 Die Phillips 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.

6.10 Die Sanyo-Schnittstelle (‘sjcd=’)

Diese Karte hat folgende Syntax:

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

6.11 Die SoundBlaster Pro-Schnittstelle (‘sbpcd=’)

Diese Karte hat folgende Syntax:

```
sbpcd=iobase,type
```

wobei `type` einer der folgenden Zeichenketten ist (egal, ob Groß- oder Kleinschreibung): ‘SoundBlaster’, ‘LaserMate’ oder ‘SPEA’. Die I/O-Basisadresse ist die der CD-ROM-Schnittstelle und *nicht* die des Sound-Teils der Karte.

7 Andere Hardware-Geräte

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

7.1 Ethernet-Geräte (‘ether=’)

Unterschiedliche Treiber verwenden unterschiedliche Parameter. Ihnen allen gemeinsam ist, daß sie alle über einen IRQ, einen Wert der I/O Port-Basisadresse und einen Namen verfügen. In seiner generischsten 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 genommen. Die `param_n` Werte (falls anwendbar) haben normalerweise für jede(n) einzelne Karte/Treiber eine unterschiedliche Bedeutung. Typische `param_n` Werte werden zur Bestimmung von Dingen wie Shared Memory-Adresse, Schnittstellen-Auswahl, DMA-Channel u.ä. verwendet.

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

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

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

WICHTIGER HINWEIS FÜR DIE BENUTZER VON MODULEN: Oben genanntes Kommando wird *keine* Überprüfung nach einer zweiten Karte erzwingen, wenn der/die Treiber als ladbare Module während der Laufzeit verwendet werden (anstatt sie in den Kernel hineinkompilieren zu lassen). Die meisten Linux-Distributionen verwenden ein bloßes Kernel-Gerüst zusammen mit einer großen Auswahl an Modul-Treibern. `ether=` gilt nur für Treiber, die direkt in den Kernel kompiliert sind.

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

7.2 Der Disketten-Treiber (‘floppy=’)

Es gibt eine Fülle von Disketten-Treiber-Optionen, die alle in der Datei `README.fd` unter `linux/drivers/block` aufgelistet sind. Diese Informationen wurden direkt dieser Datei entnommen.

floppy=mask,allowed_drive_mask

Setzt die Bitmaske der möglichen Laufwerke auf `mask`. Standardmäßig sind nur die Einheiten 0 und 1 eines jeden Floppy- Kontrollers zugelassen. Dies liegt darin begründet, daß bestimmte außer-standardmäßige Hardware (ASUS PCI-Motherboards) die Tastatur durch Zugriff auf die Einheiten 2 or 3 durcheinanderbringen. Diese Option ist durch die Option `cmos` etwas veraltet.

floppy=all_drives

Bestimmt die Bitmaske der möglichen Laufwerke für alle Laufwerke. Man verwende diese Option, wenn man mehr als zwei Laufwerke an einem Floppy-Kontroller angeschlossen hat.

floppy=asus_pci

Die Bitmaske wird so eingestellt, daß sie nur die Einheiten 0 und 1 erlaubt. (Standard)

floppy=daring

Teilt den Floppy-Treiber mit, daß man einen gut funktionierenden Floppy- Kontroller hat. Dies ermöglicht ein effizienteres und glatteres Arbeiten, kann jedoch bei bestimmten Kontrollern versagen. Bestimmte Aktionen können dadurch verschnellert werden.

floppy=0,daring

Teilt dem Floppy-Treiber mit, daß der Floppy-Kontroller mit Vorsicht behandelt werden soll.

floppy=one_fdc

Teilt dem Floppy-Treiber mit, daß nur Floppy-Kontroller vorhanden sind (Standard)

floppy=two_fdc *or* floppy=address,two_fdc

Teilt dem Floppy-Treiber mit, daß zwei Floppy-Kontroller vorhanden sind. Es wird angenommen, daß sich der zweite Floppy-Kontroller auf der Adresse <address> befindet. Ist keine Adresse angegeben, wird 0x370 angenommen.

floppy=thinkpad

Teilt dem Floppy-Treiber mit, daß ein Thinkpad vorhanden ist. Thinkpads verwenden eine umgekehrte Konvention für die Platten-Änderungszeile.

floppy=0,thinkpad

Teilt dem Floppy-Treiber mit, daß kein Thinkpad vorhanden ist.

floppy=drive,type,cmos

Setzt den cmos Typ von `drive` auf `type`. Zusätzlich ist dieses Laufwerk in der Bitmaske erlaubt. Dies ist dann hilfreich, wenn mehr als zwei Floppy-Laufwerke vorhanden sind (es können lediglich zwei im physikalischen cmos beschrieben werden) oder wenn das BIOS außer-standardmäßige CMOS-Typen verwendet. Wenn CMOS für die ersten beiden Laufwerke auf 0 gesetzt wird (Standard), dann wird der Floppy-Treiber den physikalischen cmos für diese Laufwerke lesen.

floppy=unexpected_interrupts

Gibt einen Warnhinweis aus, wenn ein unerwarteter Interrupt erhalten wird (Standardverhalten)

floppy=no_unexpected_interrupts *or* floppy=L40SX

Gibt keine Nachricht aus, wenn ein unerwarteter Interrupt erhalten wird. Dies wird auf IBM L40SX Laptops in bestimmten Grafikmodi benötigt. (Es scheint eine Interaktion zwischen Grafikkarte und Floppy zu bestehen. Die unerwarteten Interrupts betreffen nur die Performance und können beruhigt ignoriert werden.)

7.3 Der Sound-Treiber (‘sound=’)

Der Sound-Treiber kann auch Bootparameter annehmen, um die hineinkompilierten Werte zu übergehen. Dies wird jedoch nicht empfohlen, da es ziemlich komplex ist. Es ist (war?) in der Datei `Readme.Linux` unter `linux/drivers/sound` beschrieben. Ein Bootparameter folgender Form wird akzeptiert:

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

wobei jeder `deviceN` Wert von folgendem Format ist `0xTaaaId` und die Bytes folgendermaßen verwendet werden:

- `T` - Geräte-Typ: 1=FM, 2=SB, 3=PAS, 4=GUS, 5=MPU401, 6=SB16, 7=SB16-MPU401
- `aaa` - I/O Adresse in hex.
- `I` - Interrupt-Zeile in hex (i.e 10=a, 11=b, ...)
- `d` - DMA-Channel.

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

7.4 Der Bus-Maus-Treiber (‘bmouse=’)

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

7.5 Der MS-Bus-Maus-Treiber (‘msmouse=’)

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

7.6 Der Druckertreiber (‘lp=’)

Bei den Kernelversionen nach 1.3.75 kann man dem Druckertreiber mitteilen, 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 (z.B. PLIP, PPA) sie stattdessen verwenden können.

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

7.7 Der ICN ISDN Treiber (‘icn=’)

Dieser ISDN-Treiber erwartet einen Bootparameter folgender Form:

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

wobei `iobase` die I/O Port-Adresse der Karte ist, `membase` die Gemeinschaftsspeicher-Base-Adresse der Karte und die zwei `icn_id` sind einzelne ASCII-Zeichenketten-Identifier.

7.8 Der PCBIT ISDN-Treiber (‘pcbit=’)

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

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

wobei `membaseN` die Shared Memory-Basisadresse der Nten Karte ist, und `irqN` die Interrupt-Einstellung der Nten Karte. Default ist IRQ 5 und membase `0xD0000`.

7.9 Der Teles ISDN-Treiber (‘teles=’)

Dieser ISDN-Treiber erwartet einen Bootparameter folgender Form:

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

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

7.10 Der DigiBoard-Treiber (‘digi=’)

Der DigiBoard-Treiber akzeptiert eine Zeichenkette von 6 durch Kommas getrennten Bezeichner oder Ganzzahlen. Hier die 6 Werte in Reihenfolge:

```
Aktivieren/Deaktivieren der Karte PC/Xi(0), PC/Xe(1),
PC/Xeve(2), PC/Xem(3) : Kartentyp wechselnde Pin-Anordnung
aktivieren/deaktivieren Anzahl der Ports auf dieser Karte I/O
Port mit konfigurierter Karte (in HEX, falls String-Bezeichner
verwendet werden) Basisadresse des Memory Window (in HEX, falls
String-Bezeichner verwendet werden)
```

Hier ein Beispiel eines korrekten Bootprompt-Parameters (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 eine I/O von `0x200` und eine Shared Memory-Basisadresse von `0xD0000` voreingestellt hat, falls kein `digi=` Boot-Argument angegeben wurde. Es wird keine automatische Hardwareerkennung durchgeführt. Weitere Informationen findet man in der Datei `linux/Documentation/digiboard.txt`.

7.11 Der RISCom/8 Multiport Serial-Treiber (‘riscom8=’)

Bis zu 4 Karten werden unterstützt, indem man 4 eindeutige I/O Port-Werte für jede einzelne Karte angibt. Weitere Informationen findet man in der Datei `linux/Documentation/riscom8.txt`.

7.12 Das Baycom Serial/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 Verwendung von Hardware-DCD, und `options=1` bedeutet Verwendung von Software-DCD. `io` und `irq` sind wie gewöhnlich die I/O Port-Basisadresse- und Interrupt-Einstellungen. Weitere Informationen findet man in der Datei `README.baycom`, die sich zur Zeit im Verzeichnis `/linux/drivers/char/` befindet.

8 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 macht nicht viel Mühe, den Text durchzugehen.

Danke,

Paul Gortmaker, `gpg109@rsphy1.anu.edu.au`