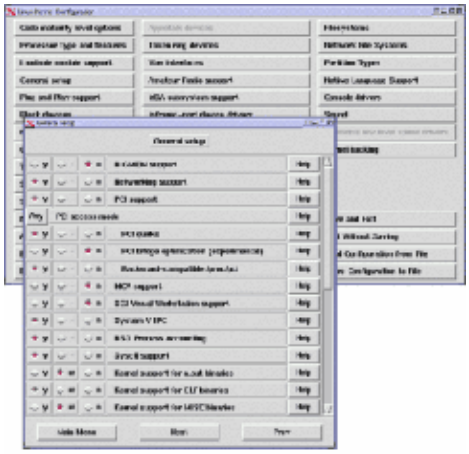
by Philip de Groot
<philipg(at)authors.linuxfocus.org>

About the author:

Philip arbeitet zur Zeit an seiner Doktorarbeit an der Universität Nimwegen und schließt sie voraussichtlich dieses Jahr ab. Thema seiner Arbeit sind sogenannte Chemometrics. Momentan arbeitet er an der Universitätsklinik Amsterdam, im Bereich Bio−Informatik. Daneben betreut er seine eigene Linux Einsteigerseiten (in Niederländisch), eines der vielen Projekte von der und für die Linux Gemeinde. Philip ist überzeugter Anwender des Betriebssystems im Zeichen des kleinen Pinguins und teilt regelmäßig seine Erfahrungen mit anderen.

Translated to English by:
Nino R. Pereira
<pereira(at)speakeasy.org>

Übersetzen des Linux−Kernels



Abstract:

Ziel dieses Artikels ist es, dem normalen Linux Anwender die Konfiguration, Übersetzung und Installation eines eigenen Linux Kernels zu vermitteln.

Einleitung

Weshalb sollte überhaupt jemand einen neuen Kernel kompilieren und installieren? Hier ein paar Gründe:

- Bessere Hardwareunterstützuung durch die neue Version.
- Der neue Kernel bietet eventuell diverse Verbesserungen, etwa bessere Unterstützung von Mehrprozessorsystemen oder USB Unterstützung. Dies gilt etwa für die 2.4.x Serie.

- Bekannte Fehler aus älteren Kernelversion wurden behoben.
- Der eigene Kernel besitzt nicht diese schicken überflüssigen Features der neuen Version und läuft deswegen zu schnell und zu stabil.

Ohne Frage verlangt die Übersetzung des Linux Kernels einiges an Erfahrungen in Sachen Computer. Dies hält die meisten Linux Einsteiger davon ab, sich damit zu beschäftigen. Der Weg zum eigenen Kernel wird in diesem Artikel mit Bildschirmfotos illustriert. Übersetzt wird der Kernel mit dem Befehl `make xconfig`, wodurch der Anwender mittels einer graphischen Oberfläche durch den Konfigurationsprozess geführt wird. Mehr als 40 Abbildungen sollen dabei helfen zu erläutern, warum man wann bestimmte Einstellungen fallweise auswählt oder eben nicht. Dies mag ein wenig übertrieben wirken, jedoch stellte diese Visualisierung den besten Weg dar, die internen Abläufe des Kernels und die Bedeutung seiner Einstellungen klar zu machen. Alle Screenshots wurden von der Konfiguration der Kernelversion 2.4.6 gemacht. Mittlerweile ist der Kernel bei Version 2.4.18 angelangt, bis auf einige zusätzliche Einträge in den Menüs (etwa für neue Hardware) unterscheidet sich der grundsätzliche Aufbau und die Vorgehensweise bei beiden Versionen nicht. Kleiner Tipp: bevor man anfängt, sollte man diesen Artikel ausdrucken, um jederzeit alle Informationen zur Hand zu haben.

Der Artikel ist wie folgt aufgebaut: Zu Beginn wird darauf eingegangen, wo die Kernel−Quellen im Internet zu finden sind und wie sie installiert werden. Danach wird die Konfiguration mittels der graphischen Benutzeroberfläche erläutert, visualisiert durch Bildschirmfotos. Nach der Konfiguration muss der Kernel übersetzt werden, doch auch ein kompilierter Kernel allein ist noch nicht verwendbar. Soll er beim Hochfahren gestartet werden, muss er noch mittels des `lilo` Bootmanagers installiert werden. Dazu wiederum ist es notwendig, die Datei `/etc/lilo.conf` anzupassen. Der Kernel kann auch auf eine Partition kopiert werden, auf welcher Linux von DOS/Windows aus mittels `loadlin` gestartet wird. Zudem gibt es eine Vielzahl spezifischer Aspekte, die berücksichtigt werden müssen, etwa PCMCIA Unterstützung, die gerade bei Laptops notwendig ist. PCMCIA Karten sind kleine, wie zu groß geratene Kreditkarten wirkende Peripheriegeräte. Eine häufige verwandte Klasse von ihnen sind Netzwerkkarten oder Modems. Diese Hardware wird vom Kernel selbst erst in der 2.4.X Serie unterstützt. Bei älteren Kernelversionen muss man zusätzliche Hilfsprogramme installieren und übersetzen. Ein weiteres Problem gibt es etwa unter SuSE Linux, da diese die ALSA Soundtreiber verwendet. Diese sind nicht Teil des regulären Kernels und müssen separat kompiliert und installiert werden, da die Originaltreiber im Allgemeinen nicht funktionieren. Um die ganze Sache noch kniffliger zu machen, kann es beim Übergang von einer Kernelserie zur anderen, etwa von 2.2.x zu 2.4.x zu Problemen mit bestimmten Kernelwerkzeugen, den sogenannten `modutils` kommen. Diese sind für den Umgang mit den Kernelmodulen zuständig. Es kann vorkommen, dass ein Kernel einer neueren Serie nichts mit der alten Version der `modutils` anfangen kann, dann muss eine neue Version dieser installiert werden. Diese Probleme treten zwar seltener auf, aber sie treten halt auf, und sie müssen einfach genannt werden.

Wer sich allerdings vertrauensvoll an diesen Artikel hält, der befindet sich prinzipiell auf der sicheren Seite. Der neue Kernel wird in `lilo` aufgenommen oder auf die entsprechende `loadlin` Partition kopiert. Dadurch kann im Fall der Fälle noch der jeweilige ursprüngliche Kernel hochgefahren werden und man kann dann daran gehen, die Störungen, die durch den Einsatz der neuen Version auftraten, zu beheben. Selbst bei Problemen mit den `modutils` unter dem neuen Kernel ist es immer noch möglich, den alten Kernel zu starten und dann etwa die neuen `modutils` separat zu kompilieren und zu installieren. Da alle Versionen dieser Werkzeuge abwärts kompatibel zu älteren Kernelversionen sind, sollten diese dann auch problemlos mit dem alten Kernel laufen.

Installation der Kernelquellen

Hierfür benötigt man `root` Rechte, also startet man im Allgemeinen damit, sich als `root` einzuloggen. Zuallererst müssen nun die Kernel Quelltexte installiert werden, etwa von CD. Bei der SuSE Distribution sind die Quellen unter "`d`" (Development, Programmentwicklung) zu finden, der Paketname lautet `lx_kernel`. Es ist durchaus anzuraten, die Kernelquellen zu installieren, die mit der eigenen Distribution mitgeliefert werden, da dadurch auch direkt die verschiedenen graphischen Oberflächen mit installiert werden. Sobald dies geschehen ist, kann das `Tar`−Archive mit der neuesten Version der Linux Kernelquellen, etwa die Datei `linux-2.4.6.tar.gz` heruntergeladen, zum Beispiel unter http://www.kernel.org/pub/linux/kernel/v2.4/ und installiert werden. Die entsprechende Version der `modutils` ist unter http://www.kernel.org/pub/linux/utils/kernel/modutils/v2.4/ zu finden. Die Versionsnummer dieser Werkzeuge muss *nicht* mit der des jeweils heruntergeladenen Kernel übereinstimmen, es ist vollkommen ausreichend, die jeweils neueste Version zu verwenden. Das Übersetzen und Installieren der `modutils` wird später unter Installation der `modutils` erläutert. Zuerst jedoch wendet man sich dem Kernel selber zu.

Die (alten) Quelltexte sind auf dem heimischen Rechner nun unter `/usr/src/linux` zu finden. Es ist ratsam, diese Quellen zu sichern, etwa durch Umbenennen:

```
cd /usr/src
mv linux linux-2.2.19 (falls die Version der eigenen
                       Kernelquellen 2.2.19 ist).
```

Erst nachdem die Original Kernelquellen umbenannt worden sind, sollte der neueste Kernelcode entpackt werden. Man stellt dabei fest, dass etwa die Datei `linux-2.4.6.tar.bz2` ihren Inhalt in das Verzeichnis `linux` standardmäßig auspackt. Sollte dieses Verzeichnis bereits existieren, wird sein Inhalt überschrieben, was zu Problemen führen kann, da dann der ursprüngliche Kernel nicht mehr übersetzt werden kann, die Konfigurationsinformationen des Originalkernels verloren sind, usw. Hier wird nach dem Entpacken des Kernelcodes das Verzeichnis `linux` in `linux-2.4.6` umbenannt und dann ein symbolischer Link `linux` auf dieses Verzeichnis angelegt. Vorteil an dieser Vorgehensweise ist, dass man auf einen Blick sehen kann, welche Version des Kernels die gerade verwendete ist. Ausserdem ist die Installation neuerer Kernelversionen einfacher. Folgende Befehle müssen (nicht vergessen, als `root`) ausgeführt werden:

```
cd /usr/src
cp ~/linux-2.4.6.tar.bz2 ( sollte dies das gewählte )
                         (TAR-Archiv sein und dieses)
                         ( ins Heimatverzeichnis '~' )
                         (heruntergeladen worden sein)
bzip2 -d linux-2.4.6.tar.bz2 (das kann was dauern )
tar -xvf linux-2.4.6.tar
mv linux linux-2.4.6
ln -s /usr/src/linux-2.4.6 /usr/src/linux
```

Ist dies erledigt, gibt man nun folgendes ein:

```
cd /usr/src/linux
make xconfig (siehe Abbildung 1)
```

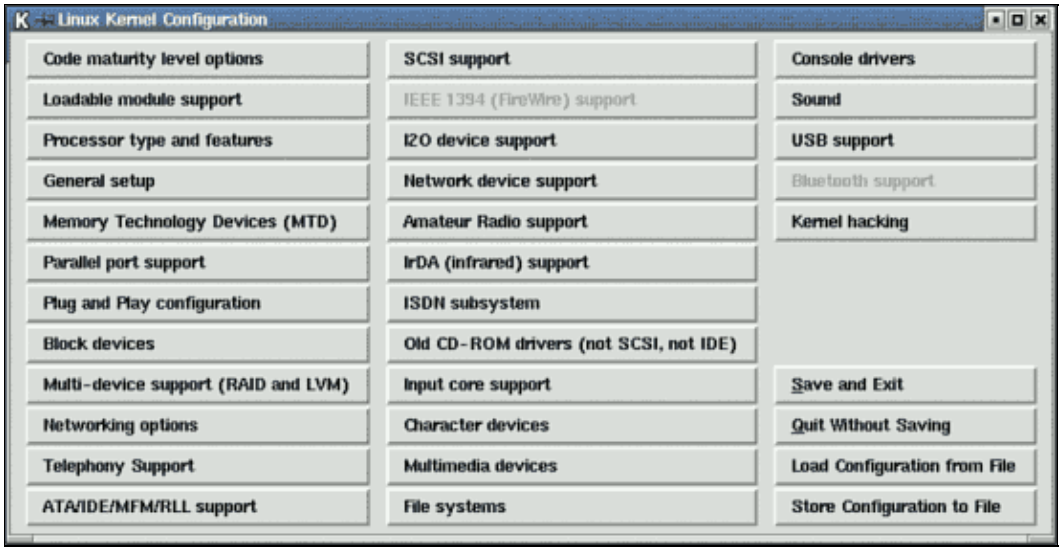


Abbildung 1: Die grafische Oberfläche zur Konfiguration des eigenen Kernels, nach Eingabe von `make xconfig`

Man findet sich im Hauptmenü der Kernelkonfiguration wieder. Die Einstellungen werden mittels Auswahl verschiedener Optionen mit der Maus gewählt. Durch Anklicken von Save and Exit werden die Einstellungen auf Platte gesichert, danach kann dann endlich der Kernel kompiliert und installiert werden (siehe Abbildung 40). Doch noch ist es nicht soweit.

Konfigurieren des Kernels

Im folgenden werden die diversen Kerneleinstellungen beispielhaft anhand von Bildschirmfotos veranschaulicht. Zu jeder Abbildung gibt es desweiteren eine Erklärung für die jeweils gewählten Einstellungen. Wer sich diese Beispiele aufmerksam durchliest, sollte keine Probleme haben, zu verstehen, warum die Wahl auf die jeweils aktivierten Optionen fiel. Zusätzlich dürfte es dadurch auch nicht allzu schwierig sein, herauszufinden, welche Einstellungen man bei der Konfiguration des eigenen Kernels wählen muss. Ähnliche Informationen über diverse Optionen kann man auch über die Hilfsfunktion während der Konfiguration mittels `make xconfig` erhalten. Dazu wählt man den `Help` Knopf und der angezeigte Text gibt dann meist einen Hinweis, welche Einstellung zu selektieren ist.

Es liegt in der Natur der Sache, dass die Beispiele hier nicht alle möglichen Geräte und Hardwarekonstellationen abdecken können. Jedoch sollten sie aufzeigen, wie bestimmte Komponenten behandelt werden müssen, und wie man nach etwaiger Unterstützung der eigenen Hardware im Kernel sucht.

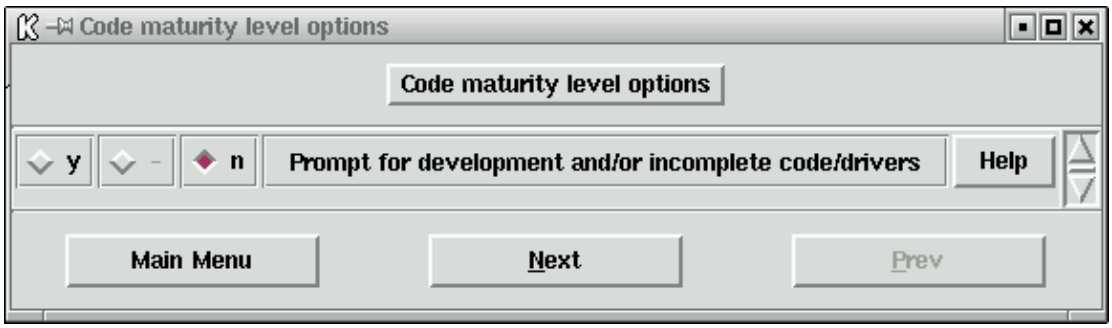


Abbildung 2: Wahl der Einstellung 'code maturity level options'.

Hier kann man einstellen, ob die noch experimentellen Optionen des Kernels zur Auswahl stehen sollen oder nicht. Manchmal mag dies notwendig sein, zum Beispiel für die Unterstützung einer relativ neuen Steckkarte. In den meisten Fällen aber sollte man diese Option deaktivieren, da der experimentelle Code im allgemeinen weniger stabil läuft. In Abbildung 1 zum Beispiel sind die Optionen *IEEE 1394 (FireWire) support* und *Bluetooth support* nicht auswählbar, da der experimentelle Code deaktiviert ist und beide Treiber noch nicht als stabil angesehen werden.

Abbildung 3: Unterstützung von Kernelmodulen.
(im folgenden werden die Abbildungen nicht mehr direkt angezeigt, sondern können bei Bedarf über die jeweiligen Links angezeigt werden)
Module sind Teile des Kernelcodes, zum Beispiel Gerätetreiber, die unabhängig vom Kernel selber, idealerweise aber gleichzeitig mit ihm, übersetzt werden. Dadurch sind sie nicht in diesen fest integriert, sondern können vielmehr bei Bedarf nachgeladen werden. Allgemein gilt, dass, wo möglich, der Kernel modular kompiliert werden sollte, da man so einen kompakten und stabilen Kernel bekommt. *Achtung:* Niemals das Dateisystem als Modul kompilieren, wie in Abbildung 32 zu sehen ist. Durch diesen Fehler kann es später passieren, dass der Kernel nicht in der Lage ist, das Root−Dateisystem zu lesen, was natürlich grundlegende Voraussetzung für das erfolgreiche Hochfahren des Linuxsystemes ist. Wie der Leser sehen wird, setze ich weniger auf Module, sondern auf Integration des Codes direkt in den Kernel, da mein Kernel so direkt und ohne Module laden zu müssen mit der Hardware kommunizieren kann. Die Entscheidung ist aber reine Geschmackssache.

Abbildung 4: Auswahl des Prozessor−Types und Einstellungen.
Hier kann der Benutzer die Art seiner CPU festlegen und verschiedene Optionen einstellen. Im Allgemeinen sind die `/dev/cpu` Optionen eher etwas für Experten, die meisten Benutzer sollten sie nicht anwählen. Wer mehr als 1 Gigabyte an Hauptspeicher (RAM, nicht Festplatte) sein eigen nennt, muss die Einstellung `High Memory Support` selektieren. Da die meisten Computer aber zwischen 64 und 512 MB Hauptspeicher (und 8 bis 60 GB Festplattenkapazität) haben, ist dies in den meisten Fällen nicht notwendig. Wessen Linux auf einem 386er oder 486SX System läuft, der muss die Option `Math Emulation` aktivieren, da diese älteren Systeme ohne mathematischen Coprozessor daherkommen, welcher aber von Linux, besser gesagt dessen Emulation, gebraucht wird. So ziemlich alle heutigen Prozessoren haben aber einen eingebauten Coprozessor, die meisten Anwender können deswegen wohl diese Option ignorieren. Der Menüpunkt `MTRR` erlaubt eine schnellere Kommunikation über den PCI oder AGP Bus. Da so ziemlich alle moderneren Systeme zumindest eine PCI, wenn nicht eine AGP Grafikkarte haben, sollte diese Option ausgewählt werden. Eigentlich kann man diese Option immer auswählen, auch wenn man keinen dieser Bustypen in seinem Rechner hat, da dadurch kein Schaden angerichtet werden kann. `Symmetric multi-processing support (SMB)` ist nur für die Anwender interessant, die eine Hauptplatine mit mehr als einem Prozessor besitzen, etwa mit 2 Pentium II CPUs. `SMP` sorgt für eine optimale Auslastung beider Prozessoren durch den Kernel. Die letzte Option `APICM` betrifft zumeist ebenfalls Mehrprozessorsysteme, kann also im Allgemeinfall ebenfalls aussen vor gelassen werden.

Abbildung 5: Allgemeine Optionen des Kernels.
Der Anwender kann hier allgemeine Einstellungen des Kernels bestimmen. Die Netzwerkunterstützung sollte immer aktiviert werden, da sie in jedem Falle, beispielsweise für den Zugang zum Internet, gebraucht wird. Linux ist stark auf das Internet ausgerichtet und kann nicht ordentlich ohne Netzwerkunterstützung arbeiten. Desweiteren wird die Netzwerkunterstützung auch unter diversen anderen Aspekten benötigt, die auf den ersten Blick eigentlich nicht viel mit Netzwerken zu tun haben. Es kann sogar passieren, dass die Kompilation des Kernels ohne Netzwerkunterstützung nicht erfolgreich ist. Kurz, der Benutzer sollte diese Option unbedingt selektieren. Jedes neuere PC−System verfügt über den PCI Bus, diese Punkte sollten deswegen ebenfalls ausgewählt werden. Die Unterstützung von `PCMCIA/CardBus` ist nicht anwählbar, da vorher ja die Verwendung von experimentellem Code ausgeschlossen wurde (siehe Abbildung 2). Laptop Besitzer werden diese Option allerdings selektieren müssen, um Netzwerk− oder Modemkarten betreiben zu können (siehe *PCMCIA support (laptops)*). `System V IPC` erlaubt es Programmen, untereinander zu

kommunizieren und sich zu synchronisieren, `Sysctl support`, dass Programme bestimmte Kerneleinstellungen ändern können, ohne dass dieser neu übersetzt werden oder das System neu gestartet werden muss. Diese Punkte sollten normalerweise aktiviert sein. Aktuelle Linux Distributionen haben als `kernel core (/proc/kcore/) format` ELF, welches das Standarddateiformat diverser Systembibliotheken ist, zum Beispiel für Codefragmente für das System und verwendet von Programmen. `ELF` ist der Nachfolger des veralteten `a.out` Formates und ähnlich den Windows `.dll` Dateien. Alle neueren Linuxprogramme verwenden ELF Bibliotheken, leider aber benötigen ältere Programme die Unterstützung des `a.out` Formates. Als Beispiel sei hier *Word Perfect 8 für das X−Window System* genannt. Dieses Linux Programm ist nur im `a.out` Format verfügbar, weswegen `xwp` schlicht und ergreifend ohne `a.out` Unterstützung den Dienst versagt. Dieses sollte man also als Modul übersetzen, plant man Programme wie `xwp` zu verwenden. Ich persönlich hab dies noch für `MISC` gemacht. Prinzipiell ist dies nicht notwendig, allerdings ist es ganz praktisch, dieses Modul zur Verfügung zu haben, verwendet man des öfteren `Java`, `Python` oder den DOS Emulator DOSEMU. Hier wurden zusätzlich noch `Power Management Support` und `Advanced Power Management BIOS Support` selektiert (nicht gezeigt in Abbildung 5). Beide Einstellungen werden für moderne ATX Boards gebraucht, soll der Kernel in der Lage sein, den Rechner beim Herunterfahren automatisch auch auszuschalten. Die anderen Stromsparfunktionen wurden nicht selektiert, da sie im Allgemeinen nicht unter X−Windows (welches bei mir standardmäßig verwendet wird) funktionieren. KDE und Gnome kommen mit eigenen Funktionen zum Power Management daher, die dann verwendet werden können.

Abbildung 6: Unterstützung von verschiedenen Speichertechnologien.
Diese Option muss zum Beispiel von Besitzern von Flashcards verwendet werden, wollen sie sie unter Linux verwenden. Flashcards kommen häufig im Bereich der Digitalfotographie zum Einsatz. Wird diese Einstellung aktiviert, so können diese Karten unter Linux, mit der entsprechenden Hardware ausgelesen und etwa Fotos als JPEG Dateien auf Festplatte kopiert werden. Wer sich nicht sicher ist, ob er sie braucht, sollte die Option nicht anwählen, dies kann bei Bedarf im Nachhinein noch nachgeholt werden.

Abbildung 7: Einstellungen des Parallelportes.
Vor der grossflächigen Verbreitung von USB war der Parallelport die Standardschnittstelle für Drucker und Scanner. Wer einen Drucker mit Parallelport−Anschluss sein eigen nennt, der muss diesen unter Linux aktivieren. Es sei hier angemerkt, dass die Konfiguration von Parallelport und Drucken zwei verschiedene Paar Schuhe sind, letzteres wird später erklärt und ist in Abbildung 28 zu sehen.

Abbildung 8: Plug & Play.
Beinahe jeder besitzt heute ein "Plug & Play" System, weswegen die meisten diesen Punkt aktivieren werden. Der Linux Kernel ist dann in der Lage, "Plug & Play" Karten zu konfigurieren und dem System zugänglich zu machen. Manchmal kann es noch notwendig werden, die Option *"Plug & Play"* im BIOS zu aktivieren, da andernfalls Linux (aber auch Windows!) die Karten nicht konfigurieren kann. *ISA Plug & Play& Support* dient der Unterstützung von PNP Karten, die noch in einem ISA Bus Steckplatz stecken. Als Beispiel sei hier der Soundblaster AWE64 angeführt. Für den ISA Bus wurde nie ein PNP Standard spezifiziert, wodurch die Konfigurierung solcher Karten recht schwierig wird. Vor der Kernelserie 2.4.x mussten Linux Anwender das Programm `isapnp` für die Konfiguration während des Hochfahrens verwenden (zu finden im Paket `isapnptools`, mit `rpm −qil isapnptools` werden alle dazugehörigen Dateien angezeigt). `isapnp` liest die Datei `/etc/isapnp.conf` aus, in welcher alle Portadressen und Interrupts der verschiedenen Karten stehen. Waren diese Informationen nicht korrekt oder wurde `isapnp` gar nicht erst verwendet, waren die entsprechenden PNP Karten nicht für das System verfügbar, Modem−, Netzwerk− oder Soundkarte funktionierten nicht. Die Option *ISA Plug & Play& Support* ersetzt diese Prozedur. Statt auf die Konfigurationsdatei `/etc/isapnp.conf` zuzugreifen, werden die Einstellungen automatisch ermittelt. Wer SuSE 7.1 verwendet, muss die Datei `/etc/isapnp.conf` umbenennen, etwa in `/etc/isapnp.conf.old` nach der Übersetzung von Kernel 2.4.x. Sowohl der Kernel, als auch `isapnp` beanspruchen dieselben Ressourcen, was katastrophale Folgen nach sich zieht. Das Problem liegt darin, dass SuSE 7.1 (und auch ältere Versionen) automatisch `isapnp` während des Bootvorganges aufrufen, selbst

wenn der Kernel seine eigenen Routinen für die Handhabung von PNP Karten mitbringt. Dies betrifft aber nur ältere Linuxsysteme, neuere verwenden standardmäßig nicht mehr `isapnp`.

Abbildung 9: Konfiguration von blockorientierten Geräten.
So ziemlich jeder wird ein Diskettenlaufwerk besitzen und dieses verwenden wollen, also wird der erste Unterpunkt selektiert, entweder als Teil des Kernels oder auch als Modul. In diesem Fall wird beim ersten Zugriff auf das Laufwerk automatisch das Modul eingebunden, sofern die Datei `/etc/conf.modules` bzw. `/etc/modules.conf` ordnungsgemäß aufgesetzt ist, was bei den meisten Distributionen der Fall ist. Der Anwender sollte keine Probleme damit haben, sollten die richtigen Einstellungen, die in Abbildung 3 zu sehen sind, gewählt worden sein. Natürlich muss für die Verwendung von Disketten noch die Unterstützung des Kernels für das entsprechende Dateisystem aktiviert werden, was in Abbildung 32 visualisiert wird.
Die anderen Optionen für Blockgeräte betreffen IDE Laufwerke, die über den Parallelport angesteuert werden sollen. Dies wird seltener gebraucht und ist in der Regel nicht aktiviert. Eine mögliche Ausnahme stellt der Punkt *loopback device support* dar. Wer unter Linux CDs brennt, erstellt normalerweise vorher ein Image der CD. Die *loopback device* wird benötigt, um sich beispielsweise den Inhalt des Images ansehen zu können.

Abbildung 10: Unterstützung von *multiple devices*.
Im Allgemeinen wird der normale Linux Anwender weder Unterstützung von RAID oder LVM Systemen benötigen. *RAID* Systeme verwenden zwei oder mehr Festplatten, um Daten parallel zu speichern. Sollte eines der Geräte einen Defekt erleiden, die anderen aber einwandfrei funktionieren, so kann das System davon unbeeinträchtigt weiter arbeiten. *LVM* hingegen ermöglicht es, durch Hinzufügen eines weiteren Gerätes eine existierende Partition zu vergrößern. Dies hat den Vorteil, dass die bestehende Partitionierung weder geändert, noch umständliches Kopieren von Daten notwendig wird. Pfadnamen bleiben dabei erhalten. Auch wenn diese Option wirklich nützlich ist, wird sie von den meisten Benutzern nicht benötigt.

Abbildung 11: Netzwerkeinstellungen.
Die Option *Packet Socket* ist notwendig für die Kommunikation von netzwerkorientierten Komponenten, ohne dass ein Netzwerkprotokoll im Kernel realisiert ist. Kurz gesagt: Diese Option sollte immer selektiert werden. Die meisten anderen hingegen bleiben deaktiviert, es sei denn, man benötigt einige von ihnen explizit. Beispielsweise kann *Network packet filtering (replaces ipchains)* gewählt werden, verwendet man etwa die SuSE Standardfirewall. Eine Firewall sichert den Rechner gegen Angriffe von ausserhalb, etwa über das Internet, zumindest, wenn sie entsprechend aufgesetzt und konfiguriert worden ist. Solch ein Sicherheitsmechanismus auf Kernelebene hat also offensichtliche Vorteile. Weitere Möglichkeiten des *network packet filtering* werden in Abbildung 12 gezeigt. Für Netzwerkverbindungen werden *Unix domain sockets* benötigt. Aber auch andere Anwendungen, etwa *X−Windows* erwartet und verwendet die Unterstützung von Unix Sockets. Deswegen sollte dieser Punkt immer ausgewählt werden. *TCP/IP networking* beinhaltet alle Protokolle, die für die Kommunikation im Internet und in eigenen internen Netzwerken benötigt werden. Es macht also normalerweise Sinn, die TCP/IP Unterstützung zu aktivieren. Besteht Unsicherheit, welche jeweilige Option notwendig ist, kann man die Hilfstexte zu Rate ziehen. Sollte danach noch Zweifel bestehen, so kann man durchaus erstmal die Punkte auswählen und sie später, im Verlaufe von Tests, immer noch, falls sie nicht notwendig sind, wieder entfernen. Hierbei ist der Einsatz von Modulen recht hilfreich.

Abbildung 12: IP Netzwerkfilter (Firewall).
Die Firewall unter SuSE Linux benötigt die Abwärtskompatibilität zu `ipchains`, weswegen dieser Punkt selektiert werden muss. Sollte eine andere Firewall oder Linux Distribution verwendet werden, so sollte man die jeweilige Dokumentation konsultieren.

Abbildung 13: Konfiguration der Unterstützung für Netzwerk−Telefonie.
Wer eine entsprechende Karte für das Telefonieren über das Internet besitzt, braucht diese Option, alle anderen und wohl die Mehrheit der Anwender hingegen kann sie dekativieren.

Abbildung 14: Unterstützung von ATA, IDE, MFM und RLL Geräten (Festplatte) .
Diese Option benötigt eigentlich so ziemlich jeder Anwender. Ausnahme hier sind diejenigen, die reine SCSI Systeme ihr eigen nennen. Deswegen werden die meisten hier diesen Punkt anwählen wollen. Selektieren der Zeile darunter bringt ein Untermenü zutage, welches eine Reihe von weiteren Optionen anbietet. Da sie recht wichtig sind, werden mehrere Bilder zur Illustration verwendet. Bei der Anwahl der Punkte sollte man recht aufmerksam vorgehen, da sie ausgesprochen wichtig sind.

Abbildung 15: ATA, IDE, MFM und RLL Unterstützung – Bild 1.
Die erste Option wird für den Betrieb jeglicher Geräte benötigt, die über eine IDE/ATAPI Schnittstelle verfügt. Darunter fallen nicht nur Festplatten, sondern auch CD−ROMs und Brenner, sowie Band− und ZIP−Laufwerke. Im Prinzip besitzen alle aktuellen PCs IDE/ATAPI Controller, weswegen die Option aktiviert werden sollte. Für die Verwendung von Festplatten muss die Option *include IDE/ATA−2 DISK support* aktiviert werden, es sei denn, man hat nur SCSI Laufwerke.

Abbildung 16: ATA, IDE, MFM und RLL Unterstüzung – Bild 2.
Für den Betrieb von ATAPI CD−Roms muss die Option *include IDE/ATAPI CDROM support* aktiviert werden. Wer aber einen ATAPI CD Brenner verwenden will, kann dies nur über die SCSI Emulation bewerkstelligen. Durch diese können CD−Roms und Brenner betrieben werden, es kann aber zu Problemen beim Mounten von CDs kommen, etwa Fehlermeldungen beim Mounten oder beim Abspielen von Audio CDs. Am besten verwendet man beide Einstellungen, *include IDE/ATAPI CDROM support* und *SCSI emulation support*, wie in Abbildung 16 gezeigt wird. Das Gerät, das die SCSI Emulation benötigt, im Allgemeinen der CD Brenner, kann dann in der Datei `/etc/lilo.conf` angegeben werden, durch Hinzufügen der Zeile `"append=hdd=ide-scsi"`. Genaueres dazu ist später unter der Rubrik *Konfiguration von LILO* zu finden. Besitzer von internen ZIP Laufwerken, die an der IDE/ATAPI Schnittstelle des Mainboards hängen, brauchen für den Betrieb des Gerätes die Einstellung *include IDE/ATAPI FLOPPY support*. Dies gilt ebenso für andere Wechselmedien, wie etwa ein LS120 Laufwerk. Für einen Großteil der Benutzer gilt, dass sie für den Betrieb ihrer Festplatten, CD−Roms und Floppys die Option *PCI IDE* aktivieren müssen. Selbiges gilt für die beiden Einstellungen für die DMA Unterstützung. DMA erlaubt der Hardware, direkt auf den Hauptspeicher zuzugreifen, ohne über den Prozessor gehen zu müssen. Dadurch werden etwa Zugriffe auf IDE Festplatten schneller, was nur im Sinne eines jeden Benutzers sein kann. Die Einstellung *sharing PCI IDE interrupts support* hingegen sollte man nicht aktivieren. Zwar können einige IDE Controller Interrupts mit anderen Karten teilen, etwa für den einwandfreien Betrieb exotischerer Netzwerkkarten, allerdings sinkt dadurch die Leistung bei der Verwendung der Festplatten, die an dem Controller hängen, der sich einen Interrupt teilt. Deswegen sollte dieser Punkt nur angewählt werden, wenn Hardwareprobleme mit Karten auf keine andere Art und Weise gelöst werden können.

Abbildung 17: ATA, IDE, MFM und RLL Unterstützung – Bild 3.
Hier wird der jeweilige Mainboard Chipsatz ausgewählt, Besitzer eines Pentium II Prozessors und einem Mainboard mit Intel Chipsatz wählen den entsprechenden Treiber aus, wie in Abbildung 17 zu sehen ist. Wer ein Board mit anderem Chipsatz hat, muss dann entsprechend einen der jeweiligen Treiber verwenden, die in der Abbildung nicht explizit zu sehen sind.

Abbildung 18: Konfiguration des SCSI Systems.
Wer SCSI Geräte verwendet, muss natürlich die Unterstützung für diese und die SCSI Controllerkarte aktivieren. In der Abbildung ist allerdings nur zu sehen, welche Einstellungen vorgenommen werden müssen, um einen ATAPI CD Brenner mittels SCSI Emulation zur Kooperation zu bewegen (Abbildung 16).

Abbildung 19: Betrieb von I2O Geräten.
Wer eine I2O Schnittstelle in seinem Rechner hat, muss diese Option anwählen. Für die meisten Benutzer ist dies allerdings nicht der Fall, sie können die Einstellung deaktiviert lassen.

Abbildung 20: Einstellungen für Netzwerkgeräte.

Es ist meist unmöglich, den Kernel ohne Netzwerkunterstützung zu übersetzen. Deswegen sollte diese Option immer selektiert werden. Ebenso sollte der Treiber für das *Dummy−Gerät* ausgewählt werden, entweder fest in den Kernel integriert oder als Modul. Linux benötigt dies zum einwandfreien Betrieb, auch wenn gar kein wirkliches Netzwerk vorhanden ist, was wohl auf die meisten Anwender zutrifft. Selbst wenn der Rechner an ein Netzwerk angeschlossen, ist greift Linux regelmäß auf das Dummy Device zu. In diesem Menü kann nun auch die Art der Netzwerkverbindung und der Treiber für etwaige Netzwerkkarten selektiert werden, wie in dem Beispiel der Abbildung 21. Modembesitzer müssen allerdings noch mehr an Einstellungen vornehmen. PPP Unterstützung, entweder mittels *PPP support for async serial ports* für Modems am seriellen Port oder *PPP support for sync tty ports*, falls man über eine schnelle Verbindung etwa über eine SyncLink Karte verfügt. Wer dies vergisst, wird vom Kernel die Fehlermeldung erhalten, dass das PPP Modul nicht existiere, auch wenn es erstellt worden ist, eine nicht wirklich hilfreiche Fehlermeldung, wenn man das Problem an sich sucht. Beide Kompressionsmethoden können problemlos gleichzeitig angewählt werden, der Kernel verwendet die richtige bei entsprechendem Bedarf.

Abbildung 21: Treiber für Ethernetkarten.
Als Beispiel wird hier eine 3COM 100 MBit Ethernetkarte mit 3c509/3c529 Chipsatz benutzt. In diesem Fall wird der Treiber als Modul übersetzt, dadurch wird er erst eingebunden, wenn er auch wirklich gebraucht wird. Natürlich muss jeder Anwender den entsprechenden Chipsatz für seine Karte aussuchen. Zusätzlich muss die Netzwerkverbindung mittels eines Konfigurationsprogrammes, wie beispielsweise `yast2` der SuSE Distribution konfiguriert werden.

Abbildung 22: Amateurfunk Unterstützung.
Diese Option wählen Anwender, die Unterstützung für Amateurfunk Hardware benötigen. Die meisten Benutzer werden diese Option allerdings nicht brauchen.

Abbildung 23: Konfiguration von Infrarot (Wireless) Kommunikation.
Notwendig für Besitzer von drahtlosen Geräten, die mittels Infrarotverbindung kommunizieren, etwa bestimmte Mäuse oder Tastaturen. Auch diese Option ist für die meisten Besitzern von Desktoprechnern uninteressant.

Abbildung 24: Konfiguration des ISDN Systemes.
Hier wird der Treiber für eine ISDN Karte im Rechner ausgesucht. Wichtig hierbei ist, den genauen Typ der Karte und ihres Chipsatzes zu kennen, da nur so der richtige Treiber selektiert werden kann.

Abbildung 25: Treiber für ältere CD−ROM Geräte.
Ältere PCs, wie 486er oder gar 386er besaßen CD−Rom Laufwerke, die nicht über die IDE Schnittstelle angeschlossen waren, sondern entweder über die Soundkarte oder mit einer eigenen Controllerkarte daher kamen. Wer solch ein CD−Rom zum Laufen bringen will, muss hier den entsprechenden Treiber auswählen. Für modernere Rechner spielen die Treiber hier jedoch keine Rolle mehr.

Abbildung 26: Konfiguration des sogenannten "Input Cores".
Hier ist eine der wichtigsten Neuerungen der 2.4.x Kernelserie zu finden, die Unterstützung für USB. Generell stellt der Input Core eine Schicht zwischen dem Kernel und einigen USB Geräten dar. Abbildung 38 zeigt die unterstützten USB Komponenten, der jeweils dazugehörige Hilfstext beschreibt, welche von ihnen diese Option hier benötigen. Alle neuen Mainboards besitzen einen USB Controller, deswegen sollte diese Option hier angewählt werden. In diesem Beispiel allerdings wurde auf USB Unterstützung verzichtet.

Abbildung 27: Zeichenorientierte Geräte – Bild 1.
Die Option *virtual terminal* gibt dem Benutzer die Möglichkeit, unter X−Windows ein `xterm` Fenster zu öffnen und die Textkonsolen zu verwenden. Im Allgemeinen wird sie immer aktiviert. Der zweite Eintrag, *support for console on virtual terminal*, legt fest, wohin der Kernel Nachrichten, wie etwa Warnungen bezüglich fehlender oder fehlerhafter Module, Kernelprobleme und Startmeldungen sendet. Unter

X−Windows wird oft ein separates Fenster für diese Art von Nachrichten verwendet, im Konsolenmodus werden sie normalerweise auf dem ersten virtuellen Terminal angezeigt (CTRL+ALT+F1). Diese Option sollte aktiviert sein. Desweiteren können Meldungen etwa an die serielle Schnittstelle, beispielsweise an einen Drucker oder ein anderes Terminal (4. Option) gesendet werden. Für das Ausdrucken der Meldungen muss noch die Schnittestelle mittels der 3. Option aktiviert werden. Selbiges gilt, sollte man eine serielle Maus verwenden. Deswegen kann man im Allgemeinen davon ausgehen, dass die Option *standard/generic (8250/16550 and compatible UARTs)* selektiert werden sollte. In diesem Beispiel wird ein Modul hierfür erstellt, da SuSE während des Hochfahrens das Fehlen des Modules `serial support` anmerkt, und auf diese Weise, dadurch dass das Modul existiert, diese Fehlermeldung vermieden wird. Die Einstellungen bezüglich der zeichenorientierten Geräte sind extrem wichtig, Fehler, die hier gemacht werden, können dazu führen, dass das System den Dienst verweigert. Deswegen wird in den Abbildungen 28−30 detaillierter auf die Optionen eingegangen.

Abbildung 28: Zeichenorientierte Geräte – Bild 2.
Soll von einem anderen Rechner aus auf dem eigenen System etwa `xterm` gestartet werden, zum Beispiel mittels `telnet` oder `ssh`, so muss die Option *unix98 PTY support* angewählt werden. Auch wenn es so aussehen mag, dass ein Einzelsystem diese Option nicht benötigt, so ist es doch so, dass eine Reihe von Hintergrundprozessen sich auf diese verlässt. Es ist deswegen ratsam, auf jeden Fall diese Einstellung zu selektieren, sei es nur, um Fehlermeldungen während des Systemstartes (zumindest bei SuSE) zu entgehen. Wer einen Drucker an seinem Rechner über den Parallelport bertreibt, der muss die Option *parallel printer support* aktivieren. Wer hingegen einen USB Drucker sein eigen nennt, braucht dies nicht zu tun. Für den Versand von Kernelnachrichten an einen Drucker am Parallelport wird die Option *Support for console on line printer* benötigt. Im Allgemeinen braucht der normale Anwender dies jedoch nicht. Die Einstellung *support for user−space parallel port device drivers* dient der Ansteuerung weiterer Arten von Geräten, die am Parallelport hängen können. Auch dies ist für den normalen Desktop Anwender nicht notwendig. Selbiges gilt für die Unterstützung von *I2C*. Manche Karten brauchen diesen Punkt für die Videoverarbeitung, sollte man allerdings feststellen, dass dies der Fall ist, so kann man die Option nachträglich immer noch hinzufügen, sobald gewährleistet ist, dass der Rest des Kernels einwandfrei funktioniert. Unterstützung von *Maus* und *Joystick* wird bei Bedarf aktiviert, allerdings verwenden nicht alle Mäuse diesen Treiber (siehe Abbildung 29). Heutzutage haben CD Brenner meist Bandlaufwerke für die Datensicherung (zumindest im Heimbereich) weitestgehend verdrängt, die wenigsten müssen deshalbe die Option *QIC−02 Tape support* aktivieren.

Abbildung 29: Zeichenorientierte Geräte – Mäuse.
Besitzer einer seriellen Maus können die Punkte hier auslassen. Alle anderen hingegen müssen sich die entsprechenden Treiber aussuchen. Wer über eine **original** Bus Maus verfügt, selektiert den obersten Punkt und wählt aus der im Anschluss erscheinenden Liste den jeweiligen Typen heraus. Die meisten heutigen Rechner verwenden einen anderen Maustypen, der im Allgemeinen (fälschlich) "Bus Maus" bzw. PS/2 Maus genannt wird. Diese Geräte sind meist unter `/dev/aux` zu finden und werden am Rechner an einen Anschluss gestöpselt, der dem für die Tastatur ähnelt. Oft werden sie auch über die Tastatur an den Rechner angeschlossen. Für den Betrieb dieser Mäse müssen die Optionen wie in Abbildung 29 angewählt werden: *mouse support (not serial and bus mice)* und *PS/2 mouse (aka "auxiliary device" support)*.

Abbildung 30: Zeichenorientierte Geräte – Bild 3.
Die Optionen für die Kernelkonfiguration zwischen Abbildung 28 und 30 werden hier nicht weiter beschrieben, sie sind normalerweise alle deaktiviert. *Ftape, the floppy tape device driver* dient dem Betrieb von Bandlaufwerken, die über den Diskettenlaufwerkscontroller angeschlossen werden. Selbst wenn man solch ein Gerät besitzt, ist es zumindest nicht beim ersten Versuch der Kernelübersetzung zwingend notwendig, die Unterstützung einzubinden. Die anderen Optionen betreffen neuere 3D Grafikkarten. Wer solch eine Karte im AGP Slot hat, schaltet die Unterstützung für *AGP* an, sowie den jeweiligen Treiber (unter */dev/agpart (AGP support)*), passend zur Karte. Es sei hier angemerkt, dass man durchaus einen funktionsfähigen Kernel erhalten kann, auch ohne diese Punkte hier zu spezifizieren, allerdings nicht notwendigerweise! Wer über ein Mainboard mit integriertem Grafikchipsatz (z.B. Intels i815) verfügt, **muss**

diese Kerneltreiber verwenden! Andernfalls wird XFree86 version 4.x, welches heute mit so ziemlich allen Distributionen mitkommt, nicht funktionieren. Besitzer von NVidia Karten, etwa mit TNT2 oder GeForce Chipsatz, werden keinen integrierten Kerneltreiber vorfinden, weswegen es für sie wenig Sinn macht, diese Optionen hier zu verwenden. Die Firma NVidia weigert sich, Hardwarespezifikationen frei verfügbar zu machen, was allerdings Voraussetzung für die Entwicklung von Open Source Kerneltreibern ist. (Anmerkung: NVidia bietet kostenlos den Download von Linux Treibern an). Auf jeden Fall können NVidia Besitzer auch ohne die Einstellungen hier XFree86 version 4.x verwenden. Die Option *Direct rendering support* dient einem Feature von XFree86, welches seit 4.0 dort zu finden ist und die Grafikleistung über Verwendung des Kernels erhöhen soll. Wer sich dies zunutze machen will, muss Besitzer einer unterstützten Grafikkarte sein und entsprechend XFree 4.x verwenden. Zusätzlich muss *AGP support* aktiviert sein. Jedoch ist es durchaus akzeptabel, diese Optionen fürs erste auszulassen und später, wenn man weiss, dass der eigene Kernel funktioniert, sie sich genauer anzusehen.

Abbildung 31: Konfiguration von Multimediageräten.
Diese Option wird für den Betrieb von Radio− und Video−Karten benötigt. Auch sie ist nicht unmittelbar für einen funktionierenden Kernel notwendig.

Abbildung 32: Einstellungen zum Dateisystem − Bild 1.
Hier kann man aussuchen, welche Dateisysteme durch den Linuxkernel unterstützt werden sollen. Wer will, kann dadurch auf Windows Festplatten und Disketten zugreifen, allerdings muss sichergestellt werden, dass ebenfalls Unterstützung für das linuxeigene *ext2* Dateisystem oder das neuere *ReiserFS* eingebunden wird. Ohne diese kann Linux nicht einmal von der eigenen Bootpartition (wie in Abbildung 3 diskutiert) starten. Für den Zugriff auf Dos/Windows Systeme muss die Option *DOS/FAT support* gewählt werden. Allerdings muss für Windows NT/2000 Festplatten ein eigenständiger Treiber, welcher nur lesenden Zugriff gestattet, verwendet werden. Dieser ist weiter unten zu finden. Desweiteren muss für das Lesen und Schreiben von Dos/Windows Geräten die Einstellung *MSDOS fs support* aktiviert werden. Die meisten Anwender werden dies wollen und sollten deswegen die Option anwählen. *VFat* wird für lange Dateinamen unter Windows 95 und 98 verwendet. Wer sowohl Windows 95/98 als auch Linux auf seinem Rechner parallel betreiben will, eine sogenannte "Dualboot" Konfiguration, wobei das jeweilig zu startende Betriebssystem mittels eines Bootlaoders, etwa LILO, beim Hochfahren gewählt wird, muss die Option *VFAT* wählen. *ISO9660* werden alle benötigen, die auf CDs in deren Standardformat zugreifen wollen. Darunter ist die Einstellung für die *Joliet extensions* zu finden, welche die Dateinamenlänge erweitert, da diese in ISO9660 auf die 8+3 Zeichen des MS−DOS Formates beschränkt ist. Beide Optionen werden heute sicherlich von so ziemlich jedem Benutzer verwendet werden. Abbildung 33 zeigt weitere Optionen, darunter auch die für das Linux Ext2 Dateisystem.

Abbildung 33: Einstellungen zum Dateisystem − Bild 2.
Die Dateien im Verzeichnis `/proc` beinhalten Informationen über den aktuellen Systemstatus, zum Beispiel die gerade belegten Interrupts. Die entsprechende Option sollte im Allgemeinen ausgewählt werden. *Second extended fs support* aktiviert Unterstützung für das unter Linux (noch) Standarddateisystem. Es ist absolut NOTWENDIG, die Unterstützung einzubinden und zwar nicht als Modul! Abbildung 32 und 33 zeigen nicht die Option für ReiserFS, welches hier ebenfalls gewählt werden kann und wohl der Nachfolger des Ext2 Dateisystemes sein wird. ReiserFS ist besser in der Lage, mit Schäden am Dateisystem umzugehen, etwa durch Stromausfall oder ähnliche Probleme. Zur Zeit ist ReiserFS immer noch in der Entwicklung und fällt deswegen unter die Kategorie "experimenteller Code". Dennoch verwenden bereits viele Distributionen dieses Dateisystem. Auch wenn ReiserFS wohl Ext2 ablösen wird, so ist es nicht wirklich für alle Partitionen, zumindest zu diesem Zeitpunkt zu empfehlen. *UDF file system support* erlaubt, auch unter Linux CDs zu lesen, die mit `packetCD` unter Windows erstellt worden sind. Diese werden mit dem UDF Dateisystem gemountet, etwa mit `mount -t udf /dev/scd0/cdrom`. In diesem Abschnitt sind auch die Netzwerkdateisysteme , sowie verschiedene Typen von Partitionen und der *Native Language Support* zu finden. Wessen Rechner nicht Teil eines größeren Netzverbundes ist, der braucht sich im Allgemeinen keine Gedanken um Netzwerk Dateisysteme machen. Alle anderen können hier die Unterstützung für das *NFS File*

System oder *SMB* aktivieren. Die Option *Partition Types* ist etwas anspruchsvoller und nicht wirklich für den korrekten Betrieb des Kernels von Nöten. Deswegen sollte sie deaktiviert werden. Abbildung 34 und 35 gehen genauer auf den *Native Language Support* ein.

Abbildung 34: Native Language Support – Bild 1.
Hier wird die Zeichentabelle ausgesucht, welche unter Linux für Dateinamen von DOS und Windows Medien verwendet werden soll. Die Tabellen in Abbildung 34 sind dabei für die normalen DOS Dateinamen zuständig. Die NLS Tabellen, die in Abbildung 35 zu sehen sind, werden für die erweiterten Dateinamen verwendet. Die erste Option in Abbildung 34 *Default NLS option* bestimmt, welche Zeichen standardmäßig unter Linux verwendet werden sollen. Abbildung 35 erklärt die Option *iso7759−15*.

Abbildung 35: Native Language Support – Bild 2.
Die Option *NKS ISO 8859−15* wird benötigt, um die Namen des Windows FAT Dateisystemes, sowie denen der Joliet Erweiterungen zum normalen CD Dateisystem korrekt darzustellen, was im allgemeinen erwünscht ist. *NLS ISO 8859−15* wird für die westlichen Sprachen verwandt und beinhaltet unter anderem das Euro Symbol. Deswegen wird fast immer diese Tabelle eingebunden. *NLS ISO 8859−1* ist der Vorgänger, beinhaltet aber nicht das Eurozeichen.

Abbildung 36: Konfiguration der Konsolentreiber.
Mit *VGA text console* wird der Textmodus in VGA Auflösung betrieben. Dies ist eine der Einstellungen, die beinahe jeder haben möchte und deswegen selektiert werden sollte. Die meisten neueren Rechner haben nicht die geringsten Probleme mit dieser Einstellung, nur bei 386ern, die ohne VGA Karte daherkommen, sieht dies anders aus. Der zweite Punkt *video mode selection support* ermöglicht es dem Benutzer, während des Hochfahrens des Systemes die Auflösung des Textmodus auszuwählen. Dies mag manchmal ganz nützlich sein, wenn man mal mehr Zeichen pro Zeile haben mag, meistens aber wird diese Option nicht verwendet. Die letzten beiden Einstellungen sind experimentell und sollten deaktiviert bleiben.

Abbildung 37: Konfiguration des Soundsystemes.
Hier werden die Einstellungen für die Soundunterstützung vorgenommen. Wer ALSA verwendet (etwa SuSE 6.3 und höher), für den reicht es hier aus, *sound card support* als **Modul** zu übersetzen. Die ALSA Treiber werden später getrennt übersetzt (mehr dazu unter *SuSE und die ALSA Soundtreiber)*. Wer dagegen die Soundtreiber des Kernels verwendet, muss den passenden Treiber zur Soundkarte wählen. So ziemlich alle Soundkartennamen sind hier aufgeführt, es sollte also keine Probleme machen, den richtigen Treiber zu finden. Funktioniert die eigene Karte bereits zufriedenstellend mit dem Standardkernel der eigenen Distribution, so kann man auch die Konfigurationsprogramme der Distribution (etwa `yast2`) verwenden, um herauszufinden, welchen Treiber man wählen muss. Gut zu wissen ist, dass die Konfiguration des Sounds nicht wirklich kritisch für die Funktionsfähigkeit des Kernels ist. Sollte man hier einen Fehler machen, so bleibt die Soundkarte zwar still, der Kernel funktioniert ansonsten aber einwandfrei.

Abbildung 38: USB Einstellungen.
Auch wenn das eigene Mainboard einen USB Anschluss hat, so mag man diesen nicht notwendigerweise verwenden. Allerdings kann es passieren, etwa mit SuSE, dass beim Deaktivieren aller USB Optionen, eine entsprechende Fehlermeldung während des Bootens erscheint. SuSE unterstützt natürlich die Verwendung von USB und versucht deswegen, das entsprechende Modul zu laden. Deswegen wird in diesem Beispiel hier Support for USB als Modul eingebunden. Auch wenn die Fehlermeldung nicht wirklich wichtig ist, so kann sie auf diese Art und Weise vermieden werden. Als Minimaleinstellungen muss man Preliminary USB device filesystem mit `y` beantworten und den jeweiligen USB Treiber auswählen. Für ältere Pentium II Mainboards ist dies UHCI (Intel PIIX4, VIA, ...), bei Boards mit neueren Intel Chipsätzen wählt man UHCI Alternate Driver (JE) support. Besitzer von Compaq Rechnern wiederum sollten OHCI support selektieren. Prinzipiell braucht man nur einen dieser drei Treiber, wer allerdings unsicher ist, wählt alle drei aus, es wird automatisch der jeweils passende geladen.

Es reicht allerdings nicht aus, den Treiber für die USB Anschlüsse des Mainboards zu aktivieren. Desweiteren müssen noch die entsprechenden Treiber(module) der Geräte, die an den Rechner angeschlossen sind, spezifiziert werden. Diese Liste ist unter *USB Device Class drivers* zu finden und bietet eine große Auswahl. Die Wahl sollte recht einfach sein, im Zweifelsfall zieht man die Hilfstexte zu Rate.

Abbildung 39: Option *kernel hacking*.
Kurz gesagt: **nicht** auswählen. Diese Einstellung ist sehr hilfreich für Programmierer, die auf der Suche nach Gründen für Kernelabstürze sind oder etwa den Cache der Festplatte auslesen wollen. Für den normalen Anwender ist sie absolut sinnlos.

Abbildung 40: Save and Exit.
So, das wars. Alles was jetzt noch zu tun ist, ist den Kernel zu übersetzen und zu installieren, wie im folgenden beschrieben wird.

Übersetzen des Kernels

Wenn man auf 'Save and Exit' klickt wird die gewählte Konfiguration in der Datei './.config' (oder '/usr/src/linux/.config', wenn man unter /usr/src/linux kompiliert) gespeichert. Es ist nützlich eine Kopie dieser Datei zu erstellen, dann hat man es leichter bei einem Kernel Upgrade z.B von 2.4.5 nach 2.4.6. Auf diese Art kann man normalerweise seine alten Einstellungen behalten und viel Arbeit sparen. Auf ähnliche Weise kann man die Konfigurationsdatei des Kernels seiner Distribution benutzen. Es kann jedoch sein, dass sich zwischen zwei Kernel−Versionen sehr viel geändert hat. In diesem Fall kann man die Datei nur begrenzt benutzen. Deshalb sollte man sich die wichtigsten Einstellungen separat aufschreiben. Die Prozedur um einen Kenel zu kompilieren ist wie folgt:

```
make dep
make clean (for older kernels)
make bzImage
make modules
make modules_install
```

In Abbildung 40 kann man schon sehen, dass der nächste Schritt 'make dep' ist. Natürlich führt man diese Befehle im Linux Quellverzeichnis aus, normalerweise '/usr/src/linux'. Kernel der 2.0.x Serie oder älter brauchen auch noch den Befehl 'make clean', der alte Dateien von einer möglicherweise vorangegangenen Kompilation löscht, bevor der neue Kernel kompiliert wird. Der 'make clean' Befehl verhindert merkwürdige Fehler, die durch alte Objektdateien (.o) verursacht werden können, die nicht ordentlich überschrieben wurden. Der Befehl 'make bzImage' kompiliert den Kernel, aber installiert ihn noch nicht. Man kann den

Kernel auch mit anderen Befehlen, wie z.B 'make bzlilo' oder 'make zImage' kompilieren, aber diese können zu Problemen führen. Die meisten Kernel sind zu groß für 'make zImage': Man bekommet eine Fehlermeldung und es wird kein Kernel gebaut. Bei 'make bzlilo' muß alles in '/etc/lilo.conf' schon korrekt konfiguriert sein, da der Kernel gleich installiert wird. Das ist aber nicht immer der Fall. Es ist daher sicherer, diese Befehle zu meiden. Der Befehl 'make modules' kompiliert die Kernel Module und 'make modules_install' installiert sie. Dieser Befehl speichert die Module unter dem Verzeichnis '/lib/modules/2.4.6/', wenn die kompilierte Kernel−Version 2.4.6 ist. Das Verzeichnis entspricht also immer der Version, die man gerade kompiliert. Auf diese Art werden alte und neue Versionen getrennt. Beim Booten weiß der Kernel welche Version er hat und nimmt die richtigen Module. Das funktioniert jedoch nicht, wenn man mehrmals denselben Kernel in unterschiedlicher Konfiguration kompiliert. Dann liegen die alten Module der gleichen Version einfach herum. Das sollte kein Problem sein, aber es ist übersichtlicher und sauberer das Verzeichis erst zu löschen oder umzubenennen, bevor man neue Module der gleichen Version installiert.

Um Probleme beim Installieren des Kernels zu vermeiden, muß man auch sicherstellen, dass die Lilokonfiguration in '/etc/lilo.conf' korrekt ist. Der Kernel und die Datei 'System.map' muß an der richtigen Stelle liegen (wie in '/etc/lilo.conf' spezifiziert). Danach muß man den Befehl 'lilo' ausführen. Eine Alternative ist 'loadlin'. Damit kann man den Linux Kernel von Dos/Windows aus booten. Beide Optionen werden im folgenden besprochen.

Lilo Konfigurieren

Die Lilokonfigurationsdatei befindet sich normalerweise im Verzeichnis '/etc' und heißt 'lilo.conf'. Öffne diese Datei in einem ASCII Editor. Wenn du einen mächtigen Editor wie XEmacs installiert hast, dann tippe 'xemacs /etc/lilo.conf &', mit kedit oder gedit
'k(g)edit /etc/lilo.conf' oder mit einem ganz einfachen Editor wie pico oder nano ('pico /etc/lilo.conf'). Die Datei wird ungefähr so aussehen.

```
boot    = /dev/hda
vga     = normal
read-only
menu-scheme = Wg:kw:Wg:Wg
lba32
prompt
timeout = 300
message = /boot/message

  other  = /dev/hda1
  label  = win98

  image  = /boot/bzImage
  label  = linux-2.4.6
  root   = /dev/hda3
  append = "parport=0x378,7 hdd=ide-scsi"

  image  = /boot/vmlinuz.suse
  label  = suse
  root   = /dev/hda3
  append = "hdd=ide-scsi"
  initrd = /boot/initrd.suse
```

Der genaue Inhalt der Datei kann bei den verschiedenen Distributionen abweichen von dem obigen Beispiel. Ich werde daher nicht jede Zeile besprechen. Der Befehl 'boot' in der ersten Zeile gibt an, von welcher physikalischen Festplatte der Bootprozess startet. 'boot' zeigt auf das sogenannte 'master boot record'. In diesem Fall boote ich von /dev/hda (erste Platte am IDE−Bus). Die Option 'vga' gibt an, dass der normale VGA Textmode mit 80x25 Zeichen benutzt wird. Die Option 'read−only' gibt an, dass die Linuxpartition erst nur−lesend gemounted wird. Während des Bootens werden die Platten auf Fehler geprüft, danach werden sie erst zum Beschreiben freigegeben. Die Zeile 'menu−scheme' gibt die Farben an, die für das Bootmenü benutzt werden. Mit der Option 'lda32' ist es möglich, ein Betriebssystem nach Zylinder 1024 zu booten, wenn das vom Bios unterstützt wird. Alle modernen Systeme unterstützen 'lba32'. Probleme mit dieser Option kann man durch einen BIOS Upgrade lösen. Der Befehl 'prompt' veranlasst Lilo zu warten und den Benutzer zu fragen, welches Betriebssystem gebootet werden soll. 'timeout' gibt die Anzahl der Millisekunden an, die Lilo auf Benutzereingaben wartet, bevor das erste Betriebssystem automatisch gebootet wird (oder das, was als 'default' angegeben ist, hier gibt es aber kein 'default' deshalb wird das erste genommen). Die 'message' Option gibt an, welches Bild beim Booten angezeigt wird. Bei Suse ist das Tux, der knuddelige Linux Pinguin. Man kann die Datei mit dem Befehl 'xv /boot/message' oder 'gimp /boot/message' ansehen. Beachte, dass die Datei /boot/message nicht auf älteren Systemen ohne grafischen Boot−Bildschirm existiert. Alle Optionen für 'lilo' sind in den Man−Seiten 'man lilo' und 'man lilo.conf' beschrieben.

Die anderen Optionen geben an, was je nach gewähltem Betriebssystem gemacht werden soll. Maximal kann man 16 verschiedene Systeme oder Kernel booten. Der Name des Systems wird mit 'label=' angegeben. Bei Windows muß man 'other' angeben, da angenommen wird, dass es immer auf der ersten Partition der Platte liegt. Der nächste Eintrag in der obigen Datei ist 'image=/boot/bzImage'. Das ist der Pfad zu dem Kernel der gebootet werden soll. Mein Linux root−Verzeichnis ist '/dev/hda3'. Die Zeile 'append = "parport=0x378,7 hdd=ide−scsi"' gibt dem Kernel an, was die Adresse und den Interrupt für den Parallel Port ist ((port 0x378, interrup t 7) und sagt, dass der CD Brenner 'hdd' über SCSI−emulation angesprochen werden soll. Der Name des CD Brenners ('hdd') hängst von der Position auf dem IDE Interface, an dem er angeschlossen ist, ab. Das Benutzen eines Interrupts für den Drucker beschleunigt das Drucken, ist aber nicht nötig, da Linux ohne diese Einstellung im etwas langsameren 'polling'−Mode arbeitet. Die Datei '/boot/vmlinuz.suse' ist bei mir der Standard Kernel, der bei Suse mitgeliefert wird. Man sollte ihn nie löschen, da man dann in Notfällen immer wieder auf diesen Kernel zugreifen kann. Die Zeile 'initrd = /boot/initrd.suse' gehört zu dem Suse Kernel und lädt eine sogenannte Ram−disk. Hier befinden sich Kernelmodule, die schon beim Booten gebraucht werden. Bei einem sebst gebauten Kernel ist das nicht nötig, aber eine Distribution muß eine viel größere Anzahl von Hardwarekonfigurationen unterstützen und das kann man nur mit einer Ram−disk erreichen.

Hoffentlich ist es jetzt klar, wo man den neuen Kernel speichern muß. In diesem Beispiel ist das:

```
cp /usr/src/linux/arch/i386/boot/bzImage /boot
cp /usr/src/linux/System.map /boot/System.map−2.4.6
lilo
```

Wenn du schon einen Kernel mit dem Namen 'bzImage' hast und du möchtest ihn behalten, dann kannst du ihn auch als '/boot/bzImage−2.4.6' kopieren und in /etc/lilo.conf /boot/bzImage in /boot/bzImage−2.4.6 ändern. Der Kompilevorgang erzeugt immer eine neue System.map die alle Symbole und die Konfiguration wichtiger Kernel Variablen enthält.
Der Befehl 'depmod −a' erzeugt eine Datei mit allen Abhängigkeiten der Kernelmodule. Bei den meisten Linuxdistributionen, wie auch bei Suse, wird depmod während des Bootens ausgeführt. Um die Datei /boot/System.map muß man sich jedoch selbst kümmern und sicherstellen, dass sie zum Kernel paßt.

Loadlin

Wenn man 'loadlin' benutzt, um Linux zu booten, muß man es nur nach C:\loadlin\ kopieren. Fast alle Linuxdistributionen haben 'loadlin' auf der ersten CD in dem 'dosutils' Verzeichnis. Man muß auch den neu kompilierten Kernel nach 'C:\loadlin' kopieren. Wenn man zuerst Windows98 im DOS−Mode startet, dann kann man Linux mit dem folgenden Befehl booten:

loadlin bzImage
Normalerweise sind einige Konfigurationsparameter, wie z.B die Kernel root Partition im Kernel selbst schon gespeichert und in diesem Fall ist der obige Befehl ausreichend, um Linux zu booten. Wenn man unter DOS 'loadlin | more' eingbit, erhält man einen Hilfetext mit einem Link zur 'loadlin−HowTo'. .

SuSE und die ALSA Sound Treiber

Suse benutzt die ALSA (Advanced Linux Sound Architecture) Sound Treiber. Diese haben eine bessere Qualität als die des OSS Projektes, die im Standard Kernel enthalten sind. Wenn man mit Sound unter Linux arbeiten möchte, sollte man unbedingt ALSA benutzen. Diese Treiber sind nicht Teil des (alten) Linuxkernels. Man muß sie deshalb separat kompilieren. Der Sourcecode für ALSA ist in dem 'zq' der SuSE Distribution. Installiere ALSA mit YaST oder YaST2 so, dass der Source Code im Verzeichnis '/usr/src/packages/' liegt. Zum Kompilieren und Installieren muß man folgendes machen:

```
rpm -bb /usr/src/packages/SPECS/alsa.spec
cd /usr/src/packages/BUILD/alsa/alsa-driver-<version number>/
./configure
make install
```

Die erste Zeile installiert die Sourcen und den Alsa Treiber in '/usr/src/packages/BUILD/'. Die ALSA Treiber werden leider standardmäßig nicht einkompiliert. Man muß sie separat kompilieren und von Hand installieren. Das geht mit './configure' und 'make install'. Der Befehl 'make install' kompiliert und installiert sie nach '/lib/modules/2.4.6/misc/'. Wenn man nun SuSE bootet, wird automatisch der richtige Soundtreiber geladen.

Falls Du kein SuSE hast oder eine neuere Version der ALSA Treiber möchtest, kannst du sie unter http://www.alsa−project.org finden. Das ist die Startseite des ALSA Projektes mit den neusten Nachrichten (z.B Feb 2002, Integration der ALSA Treiber in den offiziellen 2.5 Kernel). Die ALSA FAQ enthält Hinweise, wie man ALSA in zum Laufen bringt, falls es bei deiner Distribution Probleme gibt.

Pcmcia (Laptops)

Unterstützung für pcmcia ist Standard für 2.4.x Kernel. Das offizielle PCMCIA HOWTO behauptet jedoch, dass man die Kernelversion von PCMCIA nicht nutzen sollte. Bis jetzt funktioniert der PCMCIA Quellcode mit allen Kernel Versionen: 2.0, 2.2, und 2.4. Meine Erfahrung ist, dass PCMCIA Treiber nicht mehr funktionieren, wenn man den Kernel neu kompiliert. PCMCIA muß man dann auch nochmal kompilieren. Entweder kann man den PCMCIA Quellcode, der in der Distribution enthalten ist, nehmen oder man bekommt ihn unter http://sourceforge.net/projects/pcmcia−cs/.
Die Installation geht wie folgt:

```
cp /etc/rc.d/pcmcia /etc/rc.d/pcmcia.SuSE
cp ~/pcmcia-cs-3.1.?.tar.gz /usr/src
```

```
cd /usr/src
tar -zxf ./pcmcia-cs-3.1.?.tar.gz
make config
make all
make install
cp /etc/rc.d/pcmcia.SuSE /etc/rc.d/pcmcia
```

Die erste Zeile löst ein SuSE spezifisches Problem. Das pcmcia−Initialisierungsscript von SuSE wird von 'make install' überschrieben, was dann unter Suse nicht funktioniert. Wenn man versehentlich schon das SuSE pcmcia Script überschrieben hat, dann muß man das pcmcia Paket von Suse nochmal installieren.

Das Quellcode RPM für pcmcia installiert man unter Suse wie folgt:

```
rpm -i /cdrom/suse/zq1/pcmcia-3.1.?.spm
cd /usr/src/packages
rpm -bb ./SPECS/pcmcia-3.1.?.spec
cd /RPMS/i386/
rpm -i --force ./pcmcia-3.1.?.rpm
SuSEconfig
```

Die Befehle nehmen an, dass die CD unter /cdrom gemounted ist. 'rpm −i' installiert den Quellcode und 'rpm −bb' kompiliert. Danach kann man das RPM wie alle anderen RPMs installieren. Beachte, dass man die Option '−−force' braucht, da sonst das Program berechtigterweise sagen wird, dass 'pcmcia' schon installiert ist. Wie immer muß man das Programm SuSEconfig aufrufen, wenn man unter Suse RPMs von Hand installiert hat, um die Konfigurationsänderungen zu aktivieren. Wenn man YaST oder YaST2 benutzt, geschieht das automatisch.

Um pcmcia zu benutzen, muß man 'network support' und 'TCP/IP support' während der Kernelkonfiguration eingeschaltet haben.

'modutils' installieren

Wie schon erwähnt, benutzt der Kernel die 'modutils', um Kernel Module zu verwalten. Die wichtigsten Befehle sind:

```
insmod (installiert ein Modul),
rmmod  (de-installiert ein Modul)
lsmod  (zeigt alle installierten Module),
```

Mit 'man lsmod' kann man mehr über lsmod erfahren. Ich werde hier jetzt nicht in Details gehen.

Kompilieren und Installieren von 'modutils' ist einfach:

```
cd /usr/src
cp ~/modutils-2.4.6.tar.bz2 .  ( angenommen, dass die modutils in deinem )
                               ( home directory, '~', sind               )
bzip2 -d modutils-2.4.6.tar.bz2
tar -xvf modutils-2.4.6.tar
cd modutils-2.4.6 ( gehe in das Verzeichnis, in das sich die       )
                  (modutils entpackt haben                         )
./configure
make ( kompilieren )
```

```
make install (installieren nach  '/sbin/'  )
```

Beachte, dass hier die 'modutils' zufällig die gleiche Versionsnummer wie der Kernel haben. Das muß nicht immer sein.

Funktioniert der Kernel richtig?

Der neue Kernel ist konfiguriert, kompiliert und über Lilo installiert. Man bootet seinen Rechner und fragt sich, wie kann ich feststellen, ob der Kernel richtig funktioniert? Der Kernel schreibt jede Menge Informationen beim Booten auf den Bildschirm. Automatisch erkannte Ports, IRQ, etc..., aber auch Fehlermeldungen. Diese Meldungen sollte man genau lesen. Bei Suse ist der weitere Bootvorgang ganz einfach. Beim Booten steht auf der rechten Seite ein grünes 'done', wenn alles funktioniert und 'failed' in rot, wenn etwas nicht stimmt. Da diese Meldungen alle sehr schnell erscheinen, werden sie über dem Login−prompt zusammen gefaßt. Von der grafischen Oberfläche kommt man dorthin mit '<Ctrl>+<Alt>+F1'. Außderdem wird das unter /var/log/ in Dateien geschrieben und die ganz frühen Meldungen bekommt man mit dem Befehl 'dmesg'. Auf diese Art kann man zumindest Hinweise bekommen über das, was nicht funktioniert hat. Fehler wie 'Cannot find module' oder 'Cannot load module' zeigen normalerweise, dass man bei der Kernelkonfiguration etwas vergessen hat.

Zusammenfassung

Mit diesem Artikel in der Hand kannst du jetzt anfangen, mit dem Kernel zu experimentieren. Ich hoffe, dass ich die Hürde erniedrigt habe. Die meiste Zeit wird man mit der korrekten Konfiguration des Kernels verbringen. Beim Kompilieren kann man etwas anderes nebenher machen.

Translation information:
nl −−> −− : Philip de Groot <philipg(at)authors.linuxfocus.org>
nl −−> en: Nino R. Pereira <pereira(at)speakeasy.org>
en −−> de: Harald Radke <harryrat(at)gmx.de>

2005−01−11, generated by lfparser_pdf version 2.51