

# Kernel kompilieren

Autor: Sebastian Wolfgarten, [sebastian@wolfgarten.com](mailto:sebastian@wolfgarten.com)

Homepage: <http://www.wolfgarten.com>

Erstellungsdatum: 30. Januar 2002, 20:41 Uhr

## Einleitung:

Dieses kleine Howto soll eine schrittweise Anleitung darstellen, mit der auch Anfänger einen neuen Kernel konfigurieren, übersetzen und installieren können. Der Kernel stellt unter Linux die Schnittstelle zwischen Hardware und Software dar. Neue Features und neue Hardware werden meist in neueren Kernelversionen implementiert, weshalb sich ab und an durchaus ein Kernelupdate lohnt. Insbesondere wenn man neue Technologien (z.B. netfilter/iptables) nutzen möchte, ist bisweilen ein Update des Kernels notwendig.

## Inhalt:

Der Kernel nimmt unter Linux sehr vielfältige Funktionen wahr. Zum einen verwaltet der Kernel den Hauptspeicher, zum anderen ermöglicht er den Zugriff auf diverse Peripheriegeräte (Schnittstellen, Laufwerke etc.). Außerdem verwaltet der Kernel den Festplattenspeicher und übernimmt die Verwaltung der Programme und Prozesse. Weitere Funktionen sind u.a. die Verwaltung von Zugriffsrechten sowie die Bereitstellung von Firewall-, Routing- und allgemeinen Netzwerkfähigkeiten.

Bereits mehrfach haben mich Leute angesprochen, die gerne einmal selber einen Kernel übersetzen möchten. Deshalb habe ich mich entschlossen, dieses Howto zu schreiben. Meine ersten Gehversuche in diesem Bereich endeten meist in kleineren und größeren Katastrophen, da ich beispielsweise die Unterstützung für das von mir verwendete Dateisystem nicht in den Kernel eingebunden hatte (beliebter Anfängerfehler!). Bevor man loslegt, schreibt man sich am besten auf, welche Hardware man benutzt (Befehl: „lspci“ u.ä.) und welche Funktionen der neue Kernel haben soll.

Genug der Worte, schreiten wir nun endlich zur Tat. Zuerst müssen wir uns von der Website [www.kernel.org](http://www.kernel.org) die gewünschte Version des Kernels herunterladen. Die aktuellste Version ist 2.4.17, wobei die dritte Zahl immer den Entwicklungsstatus darstellt. Ungerade Entwicklungsstadien sollten nicht benutzt werden, deshalb verwende ich für diesen Howto den Kernel 2.4.16, der unter <http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.16.tar.gz> (knapp 25 MB) zum Download bereit steht. Durch Eingabe des Befehls "tar xvzf linux-2.4.16.tar.gz" entpacken wir den Kernel.

Zuerst sollten wir den Kernel umbenennen, um den Überblick über die verschiedenen Versionen nicht zu verlieren. Mit "mv linux linux-2.4.16" geschieht dies, danach kann der Kernel mit "cp -R linux-2.4.16 /usr/src" ins Verzeichnis /usr/src kopiert werden, wo bereits die Quellen des aktuell installierten Kernels liegen sollten. Sollte dort ein Symbolic-Link namens linux vorhanden sein, der in der Regel auf die aktuell installierte Kernelversion verweist, muss dieser durch Eingabe von "rm /usr/src/linux" gelöscht werden. Nun kann durch Eingabe von "ln -s /usr/src/linux-2.4.16 /usr/src/linux" ein neuer Link namens linux auf das neue Verzeichnis unseres 2.4.16 Kernels erzeugt werden.

Wir wechseln nun in das Verzeichnis /usr/src/linux und rufen durch Eingabe des Befehls "make menuconfig" die grafische Oberfläche zur Konfiguration der Kernelparameter auf.

Achtung: Die Bibliothek "ncurses" muss dazu installiert sein. Falls diese nicht installiert ist, müssen Sie diese nachinstallieren.

Wir befinden uns nun in einer kleinen grafischen Oberfläche zur Konfiguration der Kernelparameter:

```
Code maturity level options --->
Loadable module support --->
Processor type and features --->
General setup --->
Memory Technology Devices (MTD) --->
Parallel port support --->
Plug and Play configuration --->
Block devices --->
Multi-device support (RAID and LVM) --->
Networking options --->
Telephony Support --->
ATA/IDE/MFM/RLL support --->
SCSI support --->
Fusion MPT device support --->
I2O device support --->
Network device support --->
Amateur Radio support --->
IrDA (infrared) support --->
SDN subsystem --->
Old CD-ROM drivers (not SCSI, not IDE) --->
Input core support --->
Character devices --->
Multimedia devices --->
File systems --->
Console drivers --->
Sound --->
USB support --->
Kernel hacking --->
---
Load an Alternate Configuration File
Save Configuration to an Alternate File
```

Dort können wir nun auf recht komfortable Weise die von uns gewünschten Optionen einstellen. Wechseln wir nun zuerst in den Bereich „Code maturity level options“, um dort die Unterstützung für experimentelle und in der Entwicklung befindliche Funktionen zu aktivieren:

```
< > module capable
---
[*] Prompt for development and/or incomplete code/drivers
```

Dies ist beispielsweise für die Unterstützung bestimmter Hardware und/oder Teile der Firewallfunktionen notwendig. Ich aktiviere diese Option eigentlich immer, da ich noch nie Probleme mit experimentellen Funktionen hatte und außerdem ich bestimmte Features (u.a. Teile von IPTables) benötige, die nur dann zur Verfügung stehen. Wir verlassen diesen Bereich und wechseln in den Bereich „Loadable module support“. Dort aktivieren wir die Möglichkeit, bestimmte Teile und Funktionen des Kernels als Module zu übersetzen, d.h. erst bei Bedarf zu laden. Diese Funktionen sind bei der Konfiguration mit einem „M“ gekennzeichnet, fest in den Kernel kompilierte Funktionen sind durch ein „\*“ gekennzeichnet. Der Sinn besteht darin, nur wirklich benötigte Funktionen in den Kernel zu kompilieren, damit dieser klein und schnell wird. Falls man bestimmte Funktionen nur unregelmäßig oder selten

benutzt, kann man diese als Module übersetzen und diese bei Bedarf durch den Befehl „insmod“ laden. Ich bin der Meinung, dass man die wirklich benötigten Treiber und Funktionen fest in den Kernel einkompiliert, damit diese einem immer zur Verfügung stehen. Dies hat natürlich den Nachteil, dass man bei neuer Hardware eventuell den Kernel neu übersetzen muss.

```
[*] Enable loadable module support
[*] Set version information on all module symbols
[*] Kernel module loader
```

Im nächsten Teil „Processor type and features“ kann man den Typ des Prozessors angeben und somit spezielle Optimierungen des Kernels nutzen. Außerdem kann man dort u.a. einen mathematischen Co-Prozessor etc. aktivieren:

```
(Pentium-III/Celeron(Coppermine)) Processor family
< > Toshiba Laptop support
< > Dell Inspiron 8000 support
< > /dev/cpu/microcode - Intel IA32 CPU microcode support
< > /dev/cpu/*/msr - Model-specific register support
< > /dev/cpu/*/cpuid - CPU information support
(off) High Memory Support
[ ] Math emulation
[ ] MTRR (Memory Type Range Register) support
[*] Symmetric multi-processing support
[ ] Multiquad NUMA system
```

Im Menüpunkt „General setup“ kann man allgemeine Einstellungen vornehmen, wie beispielsweise PCMCIA-Unterstützung, Advanced Power Management und Netzwerkunterstützung etc. aktivieren bzw. deaktivieren.

```
[*] Networking support
[*] PCI support
(Any) PCI access mode
[*] PCI device name database
[ ] EISA support
[ ] MCA support
[*] Support for hot-pluggable devices
PCMCIA/CardBus support --->
PCI Hotplug Support --->
[*] System V IPC
[ ] BSD Process Accounting
[*] Sysctl support
(ELF) Kernel core (/proc/kcore) format
<*> Kernel support for a.out binaries
<*> Kernel support for ELF binaries
<*> Kernel support for MISC binaries
[*] Power Management support
[ ] ACPI support (NEW)
< > Advanced Power Management BIOS support
```

Den nächsten Menüpunkt habe ich in meiner Konfiguration ausgelassen, da ich über kein „Memory Technology Device (MTD)“ verfüge. „Parallel port“ Unterstützung habe ich im nächsten Punkt auch deaktiviert, da ich diesen nicht benötige. „Plug-and-play“ Unterstützung habe ich aktiviert, damit meine PnP-fähigen PCI Steckkarten auch korrekt funktionieren. Im Bereich „Block devices“ habe ich nur den normalen Floppy Unterstützung aktiviert, da ich die weiteren Funktionen wie Ramdisk etc. nicht benötige. Benutzt man beispielsweise das ReiserFs Dateisystem sollte man eine Ramdisk unbedingt benutzen.

```
< * > Normal PC floppy disk support
< > XT hard disk support
< > Compaq SMART2 support
< > Compaq Smart Array 5xxx support
< > Mylex DAC960/DAC1100 PCI RAID Controller support
< > Loopback device support
< > Network block device support
< > RAM disk support
```

Den Menüpunkt „Multi-device support“ habe ich komplett deaktiviert, da ich leider über kein RAID System verfüge. Bei den „networking options“ habe ich sehr viele nützliche Funktionen aktiviert, da ich viel mit iptables arbeite und auch ansonsten zahlreiche Netzwerkfunktionen benötige. Allerdings benötige ich beispielsweise das IPX Protokoll überhaupt nicht und habe es deshalb deaktiviert.

```
< * > Packet socket
[*] Packet socket: mmaped IO
[*] Kernel/User netlink socket
[*] Routing messages (NEW)
< * > Netlink device emulation (NEW)
[*] Network packet filtering (replaces ipchains)
[*] Network packet filtering debugging (NEW)
[*] Socket Filtering
< * > Unix domain sockets
[*] TCP/IP networking
[*] IP: multicasting
[*] IP: advanced router
[*] IP: policy routing (NEW)
[*] IP: use netfilter MARK value as routing key (NEW)
[*] IP: fast network address translation (NEW)
[*] IP: equal cost multipath (NEW)
[*] IP: use TOS value as routing key (NEW)
[*] IP: verbose route monitoring (NEW)
[*] IP: large routing tables (NEW)
[ ] IP: kernel level autoconfiguration
< * > IP: tunneling
< > IP: GRE tunnels over IP
[ ] IP: multicast routing
[ ] IP: ARP daemon support (EXPERIMENTAL) (NEW)
[ ] IP: TCP Explicit Congestion Notification support
[ ] IP: TCP syncookie support (disabled per default)
IP: Netfilter Configuration --->
< > The IPv6 protocol (EXPERIMENTAL) (NEW)
< > Kernel httpd acceleration (EXPERIMENTAL) (NEW)
< > Asynchronous Transfer Mode (ATM) (EXPERIMENTAL) (NEW)
< > 802.1D VLAN Support (EXPERIMENTAL) (NEW)
---
< > The IPX protocol
< > Appletalk protocol support
< > DECnet Support
< > 802.1d Ethernet Bridging
< > CCITT X.25 Packet Layer (EXPERIMENTAL) (NEW)
< > LAPB Data Link Driver (EXPERIMENTAL) (NEW)
[ ] 802.2 LLC (EXPERIMENTAL) (NEW)
[ ] Frame Diverter (EXPERIMENTAL) (NEW)
< > Acorn Econet/AUN protocols (EXPERIMENTAL) (NEW)
```

Im Bereich der „Netfilter Configuration“ habe ich folgende Optionen aktiviert:

```
<*) Connection tracking (required for masq/NAT) (NEW)
<*) FTP protocol support (NEW)
<*) IRC protocol support (NEW)
< > Userspace queuing via NETLINK (EXPERIMENTAL) (NEW)
<*) IP tables support (required for filtering/masq/NAT) (NEW)
<*) Limit match support (NEW)
<*) MAC address match support (NEW)
<*) netfilter MARK match support (NEW)
<*) Multiple port match support (NEW)
<*) TOS match support (NEW)
<*) LENGTH match support (NEW)
<*) TTL match support (NEW)
<*) tcpmss match support (NEW)
<*) Connection state match support (NEW)
<*) Unclean match support (EXPERIMENTAL) (NEW)
<*) Owner match support (EXPERIMENTAL) (NEW)
<*) Packet filtering (NEW)
<*) REJECT target support (NEW)
<*) MIRROR target support (EXPERIMENTAL) (NEW)
<*) Full NAT (NEW)
<*) MASQUERADE target support (NEW)
<*) REDIRECT target support (NEW)
< > Basic SNMP-ALG support (EXPERIMENTAL) (NEW)
<*) Packet mangling (NEW)
<*) IOS target support (NEW)
<*) MARK target support (NEW)
<*) LOG target support (NEW)
<*) TCPMSS target support (NEW)
```

Den nächsten Bereich „Telephony support“ habe ich komplett deaktiviert, da ich dieses Feature wirklich nicht brauche. Die Optionen von „ATA/IDE/MFM/RLL support“ habe ich nicht geändert, da sie mir korrekt erschienen. Im SCSI Bereich habe ich einiges aktiviert, da ich über einen SCSI Brenner sowie ein SCSI CD-Rom Laufwerk verfüge, die ich über einen Tekram Controller anspreche. Dementsprechend sind hier die Optionen wie folgt:

```
<*) SCSI support
--- SCSI support type (disk, tape, CD-ROM)
<*) SCSI disk support
(40) Maximum number of SCSI disks that can be loaded as modules
< > SCSI tape support
< > SCSI OnStream SC-x0 tape support
<*) SCSI CD-ROM support
[*] Enable vendor-specific extensions (for SCSI CDROM) (NEW)
(2) Maximum number of CDROM devices that can be loaded as modules (NEW)
<*) SCSI generic support
--- Some SCSI devices (e.g. CD jukebox) support multiple LUNs
[*] Enable extra checks in new queuing code
[*] Probe all LUNs on each SCSI device
[*] Verbose SCSI error reporting (kernel size +=12K)
[ ] SCSI logging facility
SCSI low-level drivers --->
```

Im Untermenüpunkt „SCSI low-level drivers“ habe ich die Option für die von mir benutzt Tekram DC390 PCI Karte direkt einkompiliert.

```
< > Symbios 53C416 SCSI support
<*) Tekram DC390(T) and Am53/79C974 SCSI support
[*] _omit_ support for non-DC390 adapters
< > Trantor T128/T128E/T228 SCSI support
```

Die nächsten drei Bereiche (Fusion MPT support, IEEE 1394 support und I20 device support) habe ich komplett deaktiviert, da ich diese nicht brauche. Im Bereich „network device support“ habe ich die Unterstützung für meine beiden Netzwerkkarten aktiviert sowie

PPP, da ich DSL benutze. Die weiteren Features wie 1 GB Netzwerkkarten Unterstützung, Wireless Lan und „Token Ring devices“ habe ich deaktiviert.

```
[*] Network device support
  ARCnet devices --->
  <M> Dummy net driver support
  < > Bonding driver support
  < > EQL (serial line load balancing) support
  < > Universal TUN/TAP device driver support
  < > Ethertap network tap (OBSOLETE) (NEW)
  < > General Instruments Surfboard 1000
Ethernet (10 or 100Mbit) --->
Ethernet (1000 Mbit) --->
[ ] FDDI driver support
[ ] HIPPI driver support (EXPERIMENTAL) (NEW)
<*> PPP (point-to-point protocol) support
[*] PPP multilink support (EXPERIMENTAL) (NEW)
[*] PPP filtering (NEW)
<M> PPP support for async serial ports (NEW)
<M> PPP support for sync tty ports (NEW)
<M> PPP Deflate compression (NEW)
<M> PPP BSD-Compress compression (NEW)
<*> PPP over Ethernet (EXPERIMENTAL) (NEW)
< > SLIP (serial line) support
Wireless LAN (non-hamradio) --->
Token Ring devices --->
[ ] Fibre Channel driver support
< > Red Creek Hardware VPN (EXPERIMENTAL) (NEW)
< > Traffic Shaper (EXPERIMENTAL) (NEW)
Man interfaces --->
```

Im Bereich der 10/100 MBit Netzwerkkarten sieht meine Konfiguration wie folgt aus:

```
[*] Ethernet (10 or 100Mbit)
  < > Sun Happy Meal 10/100baseT support
  < > Sun GEM support
  [ ] 3 COM cards
  < > AMD LANCE and PCnet (AT1500 and NE2100) support
  [ ] Western Digital/SMC cards
  [ ] Racal-Interlan (Micom) NI cards
  < > AT1700/1720 support (EXPERIMENTAL)
  < > DEPCA, DE10x, DE200, DE201, DE202, DE422 support
  < > HP 10/100VG PCLAN (ISA, EISA, PCI) support
  [ ] Other ISA cards
[*] EISA, VLB, PCI and on board controllers
  < > AMD PCnet32 PCI support
  < > Adaptec Starfire support (EXPERIMENTAL)
  < > Ansel Communications EISA 3200 support (EXPERIMENTAL)
  < > Apricot Xen-II on board Ethernet
  < > CS89x0 support
  < > DECchip Tulip (dc21x4x) PCI support
  < > Generic DECchip & DIGITAL EtherWORKS PCI/EISA
  < > Digi Intl. RightSwitch SE-X support
  < > Davicoa DM910x/DM980x support
  < > EtherExpressPro/100 support
  < > Myson MTD-8xx PCI Ethernet support
  < > National Semiconductor DP8381x series PCI Ethernet support
  <*> PCI NE2000 and clones support (see help)
  < > RealTek RTL-8139 C+ PCI Fast Ethernet Adapter support (EXPERIMENTAL)
  <*> RealTek RTL-8139 PCI Fast Ethernet Adapter support
  <*> Use PIO instead of MMIO
  <*> Support for automatic channel equalization (EXPERIMENTAL)
  <*> Support for older RTL-8129/8130 boards
  < > SiS 900/7016 PCI Fast Ethernet Adapter support
  < > SMC EtherPower II
  < > Sundance Alta support
  < > TI ThunderLAN support
  < > VIA Rhine support
  < > Winbond W89c840 Ethernet support
  <M> Pocket and portable adapters
```

Die nächsten fünf Optionen „Amateur Radio support, IrDA (infrared) support, ISDN subsystem, Old CD-ROM drivers (not SCSI, not IDE), Input core support“ habe ich komplett deaktiviert, da ich diese ebenfalls nicht beanspruche. Die Option „Character devices“ bietet die Unterstützung von Mäusen, Joysticks und diverse Grafikkarten und –Karten.

```

[*] Virtual terminal
[*] Support for console on virtual terminal
<*> Standard/generic (8250/16550 and compatible UARTs) serial support
[ ] Support for console on serial port
[ ] Extended dumb serial driver options
[ ] Non-standard serial port support
[*] Unix98 PTY support
(256) Maximum number of Unix98 PTYs in use (0-2048)
I2C support --->
Mice --->
Joysticks --->
< > QIC-02 tape support
Watchdog Cards --->
< > Intel i8x0 Random Number Generator support
< > /dev/nvram support
< > Enhanced Real Time Clock Support
< > Double Talk PC internal speech card support
< > Siemens R3964 line discipline
< > Applicom intelligent fieldbus card support
< > Sony Vaio Programmable I/O Control Device support (NEW)
Floppy, the floppy tape device driver --->
<*> /dev/agpgart (AGP Support)
[*] Intel 440LX/BX/GX and I815/I830M/I840/I850 support
[*] Intel I810/I815/I830M (on-board) support
[*] VIA chipset support
[*] AMD Irongate, 761, and 762 support
[*] Generic SiS support
[*] ALI chipset support
[ ] Serverworks LE/HE support
[ ] Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)
< > ACP Modem (Mwave) support

```

Im nun folgenden Bereich „Multimedia devices“ habe ich „Video for Linux“ deaktiviert, da ich an meinem Server keine Videos gucke und sowieso keine Fernsehkarte o.ä. besitze. Der Punkt „File system“ aktiviert bzw. deaktiviert wie der Name schon sagt, die Unterstützung für Dateisysteme verschiedenster Art. Hier kann man definieren, welche Dateisysteme man benutzen möchte und welche nicht. Ich habe mich hier für die nachfolgenden Optionen entschieden:

```

[*] Quota support
<*> Kernel automounter support
<*> Kernel automounter version 4 support (also supports v3)
< > Reiserfs support
< > JDFS file system support
< > Amiga FFS file system support (EXPERIMENTAL)
< > Apple Macintosh file system support (EXPERIMENTAL)
< > BFS file system support (EXPERIMENTAL)
< > Ext3 journalling file system support (EXPERIMENTAL)
<*> DOS FAT fs support
<*> MSDOS fs support
< > MSDOS: Unix-like file system on top of standard MSDOS fs
<*> VFAT (Windows-95) fs support
< > FFS file system support (read only) (EXPERIMENTAL)
< > Compressed ROM file system support
[*] Virtual memory file system support (former sha fs)
< > Simple RAM-based file system support
<*> ISO 9660 CDROM file system support
[*] Microsoft Joliet CDROM extensions
[*] Transparent decompression extension
<*> Minix fs support
< > FreeVxFS file system support (VERITAS VxFS(TM) compatible)
<*> NFS file system support (read only)
[ ] NFS write support (DANGEROUS)
< > OS/2 HPFS file system support
[*] /proc file system support
[ ] /dev file system support (EXPERIMENTAL)
[*] /dev/pts file system for Unix98 PTYs
< > CNX4 file system support (read only) (EXPERIMENTAL)
< > ROM file system support
<*> Second extended fs support
< > System V/Xenix/V7/Coherent file system support
<*> UDF file system support (read only)
< > VFS file system support (read only)
Network File Systems --->
Partition Types --->
Native Language Support --->

```

Im Unterpunkt „Network file systems“ habe ich die Unterstützung für das Samba Protokoll aktiviert:

```
< > Coda file system support (advanced network fs)
< > InterMezzo file system support (experimental, replicating fs)
< > NFS file system support
< > NFS server support
[*] SMB file system support (to mount Windows shares etc.)
[*] Use a default NLS (NEW)
    Default Remote NLS Option: "cp437" (NEW)
< > NCP file system support (to mount NetWare volumes)
```

Die „Partition Types“ habe ich nicht geändert und bei „Native Language support“ habe ich die für mich geltende Code page eingestellt:

```
Default NLS Option: "iso8859-1" (NEW)
< > Codepage 437 (United States, Canada) (NEW)
< > Codepage 737 (Greek) (NEW)
< > Codepage 775 (Baltic Rim) (NEW)
[*] Codepage 850 (Europe) (NEW)
< > Codepage 852 (Central/Eastern Europe) (NEW)
< > Codepage 855 (Cyrillic) (NEW)
< > Codepage 857 (Turkish) (NEW)
< > Codepage 860 (Portuguese) (NEW)
< > Codepage 861 (Icelandic) (NEW)
< > Codepage 862 (Hebrew) (NEW)
< > Codepage 863 (Canadian French) (NEW)
< > Codepage 864 (Arabic) (NEW)
< > Codepage 865 (Norwegian, Danish) (NEW)
< > Codepage 866 (Cyrillic/Russian) (NEW)
< > Codepage 869 (Greek) (NEW)
[*] Simplified Chinese charset (CP936, GB2312) (NEW)
< > Traditional Chinese charset (Big5) (NEW)
< > Japanese charsets (Shift-JIS, EUC-JP) (NEW)
< > Korean charset (CP949, EUC-KR) (NEW)
< > Thai charset (CP874, TIS-620) (NEW)
< > Hebrew charsets (ISO-8859-8, CP1255) (NEW)
< > Windows CP1251 (Bulgarian, Belarusian) (NEW)
[*] NLS ISO 8859-1 (Latin 1; Western European Languages) (NEW)
< > NLS ISO 8859-2 (Latin 2; Slavic/Central European Languages) (NEW)
< > NLS ISO 8859-3 (Latin 3; Esperanto, Galician, Maltese, Turkish) (NEW)
< > NLS ISO 8859-4 (Latin 4; old Baltic charset) (NEW)
< > NLS ISO 8859-5 (Cyrillic) (NEW)
< > NLS ISO 8859-6 (Arabic) (NEW)
< > NLS ISO 8859-7 (Modern Greek) (NEW)
< > NLS ISO 8859-9 (Latin 5; Turkish) (NEW)
< > NLS ISO 8859-13 (Latin 7; Baltic) (NEW)
< > NLS ISO 8859-14 (Latin 8; Celtic) (NEW)
< > NLS ISO 8859-15 (Latin 9; Western European Languages with Euro) (NEW)
< > NLS KOI8-R (Russian) (NEW)
< > NLS KOI8-U/RU (Ukrainian, Belarusian) (NEW)
< > NLS UTF8 (NEW)
```

Im Konfigurationsbereich „Console drivers“ habe ich die nachfolgenden Einstellungen vorgenommen:

```
[*] VGA text console
[*] Video mode selection support
< > MDA text console (dual-headed) (EXPERIMENTAL) (NEW)
Frame-buffer support --->
```

Die nächsten vier Optionen habe ich komplett deaktiviert, da ich diese nicht benötige. Nun habe ich diese Konfiguration gespeichert und das Programm verlassen.

Als nächsten Schritt müssen wir nun die Abhängigkeiten durch den Befehl "make dep" prüfen. Falls dieser ohne eine ersichtliche Fehlermeldung durchläuft, können überflüssige Dateien durch Eingabe von "make clean" gelöscht werden. Durch Eingabe des Befehls "make bzImage" beginnt nun die Kompilierung des Kernels. Dieser Vorgang kann unter Umständen einige Zeit dauern, bei meinem Testsystem (Intel Celeron 500, 256 MB RAM)

hat die Übersetzung des Kernels mit den hier vorgestellten Optionen etwa 25 Minuten gedauert. Wenn die Übersetzung des Kernels ist fertig und meldet sich wie folgt:

```
Root device is (3, 3)
Boot sector 512 bytes.
Setup is 2516 bytes.
System is 901 kB
make[1]: Leaving directory `/usr/src/linux-2.4.16/arch/i386/boot'
diebels:/usr/src/linux# █
```

Eventuell kann eine Meldung erscheinen, wie „Kernel too big“ o.ä., dann muss der Kernel neu übersetzt werden, wobei weitere Optionen deaktiviert werden müssen bzw. als Module übersetzt werden sollten.

Durch Eingabe von "mv arch/i386/boot/bzImage /boot/vmlinuz-2.4.16" wird der frisch übersetzte Kernel in das /boot Verzeichnis kopiert und wir können uns den letzten beiden Schritten widmen. Durch Eingabe von "make modules" werden die Module kompiliert und wenn auch dieser Vorgang ohne Fehlermeldung durchläuft, können diese Module durch Eingabe von "make modules\_install" ins /lib/modules Verzeichnis installiert werden.

Zum Schluss muss noch die /etc/lilo.conf angepasst werden, damit beim nächsten Systemstart auch der neue Kernel gebootet wird:

```
image = /boot/vmlinuz-2.4.16
root = /dev/hda3
label = linux
```

Ebenso sollte man eine Kopie des alten Kernels als Sicherung in die /etc/lilo.conf schreiben, falls der neue Kernel nicht bootet. Führen Sie nun den Befehl „lilo“ aus, um die Änderungen in den Master Boot Record zu schreiben.

Booten Sie nun den Rechner durch Eingabe von „shutdown -r now“ und verfolgen Sie die Meldungen beim Booten des Rechners. Wenn Sie die Unterstützung für das von Ihnen verwendete Dateisystem sowie für Ihre Hardware und Netzwerk richtig aktiviert haben, sollte alles laufen.

Der neue Kernel sollte Sie durch Eingabe von „uname -r“ fröhlich begrüßen:

```
diebels:/usr/src/linux# uname -r
2.4.16
diebels:/usr/src/linux# █
```

Fertig!