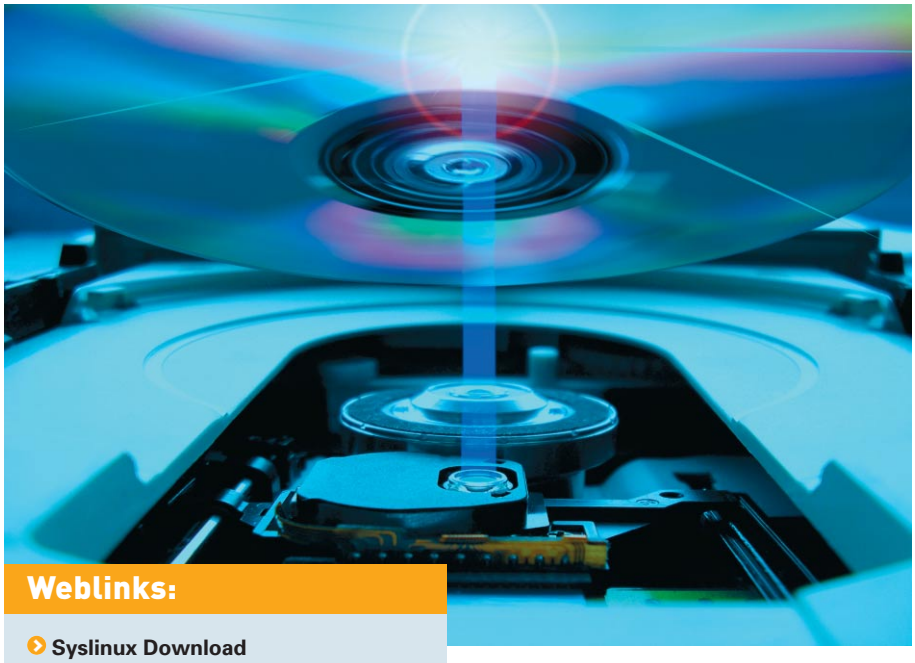
EIGENE KNOPPIX-CD

Knoppix Remastered

Knoppix eignet sich prima für viele Standardaufgaben, doch was, wenn aus Knoppix eine Demo-CD in eigenem Branding oder die eigene Virenscanner-DVD entstehen soll? Wir stellen Wege zum Remastern vor.

VON **MATTIAS SCHLENKER**



Weblinks:

- **Syslinux Download**
- **http://syslinux.zytor.org**
- **Cloop-Kernelmodul**
- **http://debian-knoppix.alioth.debian.org/sources**
- **UCK - Ubuntu Customization Kit**
- **http://uck.sf.net**
- **AUFS - Another UnionFS**
- **http://aufs.sf.net**
- **genext2fs**
- **http://genext2fs.sf.net**
- **BusyBox**
- **www.busybox.net**

Erste Linux-Live-CDs kamen Ende der 1990er auf: Besonders beliebt waren CDs in Scheckkartengröße, die auf gut 50 Megabyte die wichtigsten Rettungswerkzeuge unterbrachten. Viel Staat war mit diesen - vornehmlich auf Messen verteilten CDs - nicht zu machen, da keine grafische Oberfläche mit an Bord war und viele Funktionen wie das Netzwerk von Hand konfiguriert werden mussten. Einen echten Quantensprung stellten deshalb die ersten Knoppix-CDs am Anfang des Jahrtausends dar: Klaus Knopper nutzte erstmals ein komprimiertes Dateisystem-Image, mit dem etwa zwei Gigabyte an Daten auf einer gewöhnlichen CD untergebracht werden konnten. Knopper setzte als Basis auf Debian, während die Konkurrenz noch auf komplett aus Quellen gebaute Systeme schwor, was zu einer recht flotten Entwicklung führte. Eine weitere Neuerung war das mit Knoppix 3.7 im Jahr 2004 eingeführte *UnionFS*. Das transparent überlagernde Dateisystem sorgt dafür, dass praktisch der gesamte Inhalt der Live-CD überschrieben werden kann. So lässt sich im Live-System Software in die gewohnten Pfade installieren, ohne dass Verrenkungen mit Umgebungsvariablen notwendig sind. Dank der Debian-Basis erlaubt Knoppix so die Nachinstallation gewöhnlicher DEB-Pakete - lediglich genügend Arbeitsspeicher für die entpackten Dateien muss vorhanden sein.

Der Bootprozess

Knoppix' Bootprozess beginnt wie der jeder anderen Distribution auch: Ein Bootloader lädt den Kernel und die initiale Ramdisk. Die initiale Ramdisk besteht in der Regel aus einem Befehlsinterpreter (also einer rudimentären Shell), einem kleinen Script *linuxrc* und Kernelmodulen. Auf einem normalen Linux-System hat die Ramdisk nicht viel mehr zu tun, als die zum Mounten des Root-Dateisystems erforderlichen Treiber zu laden und schließlich den Mountpoint der Initrd mit dem Mountpoint des Root-Dateisystems zu vertauschen. Relativ schnell übergibt *linuxrc* nach dem Mounten die Kontrolle an das Programm *init* auf der Festplatte. Anders sieht es bei der Live-CD aus: Das Script *linuxrc* muss hier zunächst einmal das Laufwerk finden, auf dem die Containerdatei liegt. Bereits aufgrund der Vielzahl an Möglichkeiten (IDE-, SATA-, USB-, Firewire- oder PCMCIA-CD und DVD-Laufwerke und dazu noch zulässige Images auf USB-Stick oder Festplatte) fällt die Logik des Scriptes *linuxrc* deutlich komplexer aus.

Auch das Mounten zusätzlicher Dateisysteme erledigt Knoppix innerhalb der *Initrd* und erreicht so, dass die folgenden Prozesse bereits schreibbaren Zugriff auf fast das gesamte Dateisystem haben. Auf das „Pivotieren" des Mountpoints und die Freigabe des von der Initrd beanspruchten Arbeitsspeichers verzichten viele Live-Distributionen deshalb - die Initrd verbleibt während der gesamten Benutzung des Systems im Speicher. Auf halbwegs aktuellen PCs sollten die unnütz belegten vier bis acht MB nicht weiter ins Gewicht fallen.

„Quick and Dirty"-Remaster

Erste Gehversuche beim Knoppix-Remastering können Sie unternehmen, ohne die große Containerdatei *KNOPPIX/KNOPPIX* zu öff-

nen und anzupassen. Knoppix führt gegen Ende des Bootvorganges das Shellscript *KNOPPIX/knoppix.sh* mit Rootrechten aus - sofern dieses vorhanden ist. Im Shellscript sind Ihrer Kreativität keine Grenzen gesetzt: Sie können mittels *dpkg -i* Software installieren oder für eine Webseitendemo Datenbankserver und Webserver starten sowie die Webverzeichnisse füllen. Der Autor dieser Zeilen nutzt beispielsweise ein beim Kunden deponiertes Notfallknoppix, das den SSH-Daemon startet, Public-Key-Authentication bereithält und nach außen tunnelt, um im Falle eines nicht mehr bootfähigen PCs eingreifen zu können. Ein denkbares Szenario ist die mit einer Jukebox wie *Jajauk* aufgewertete Knoppix-DVD mit der CD-Version von Knoppix, die fast vier GByte MP3s transportieren kann. Eine solche DVD kann im Gegensatz zu einem iPod nach dem Partyrausch auch vergessen werden, ohne dass ein großer Schaden entsteht.

Die Vorgehensweise für den Bau einer derartig modifizierten Knoppix-CD ist äußerst simpel. Zunächst kopieren Sie den Inhalt einer Knoppix-CD in Ihr Arbeitsverzeichnis. Hierfür genügt es, das ISO „loopback“ zu mounten und dann mit *rsync* auf Festplatte zu kopieren:

```
mount -o loop knoppix.iso /tmp/knoppix
rsync -av /tmp/knoppix/ work
```

Vergeben Sie schließlich noch Schreibrechte auf dem Arbeitsverzeichnis:

```
chmod -R +w work
```

Und die anderen?

Neben Knoppix gibt es eine ganze Reihe weiterer Live-Distributionen, die sich mal mehr mal weniger zur Anpassung eignen. Ebenfalls auf Debian basiert *sidux*, das allerdings statt auf *Cloop* auf das *Squashfs*-Dateisystem setzt. Für die Initrd kommt das etwas modernere *initramfs*, ein komprimiertes CPIO-Archiv, zum Einsatz.

Als Bootloader wird GRUB benutzt. Da sidux auf dem stets aktuellen *Debian Unstable* aufbaut, ist aktuelle Software meist kein Problem. Eng mit Sidux verwandt, respektive dessen einstige Basis, ist *Kanotix*, dessen Entwicklung derzeit trotz der stabilen Ubuntu-Basis etwas zäh vorankommt.

Auf Slackware basiert das relativ schlanke *Slax*, das ein LZMA-komprimiertes *Squashfs* für den Container nutzt und ebenfalls auf Initramfs setzt. Die gesamte Slax-Entwicklung ist relativ offen, sodass es sich lohnt, die Entwicklung der Derivate mit den Slax-Entwicklern abzustimmen oder wenigstens deren Build-Umgebung zu nutzen.

Ubuntu ist weniger als Live-CD denn als auf Festplatte installierbare Distribution bekannt. Modifiziert man die Live-CD mit Original-Ubuntu-Paketen, erhält man eine installierbare CD oder DVD, die gewünschte Software auf einen Rutsch installiert - ideal, wenn man für Seminare und Workshops modifizierte Live-CDs erstellen möchte, mit denen die Schüler zu Hause trainieren können. Auch Ubuntu nutzt *Squashfs*, allerdings ist nach dem Anpassen des Paketumfangs und vor dem Bauen des Containers ein zusätzlicher Schritt notwendig, in dem die auf Platte verbleibenden Pakete festgelegt werden. Viele Standard-Remaster lassen sich mit dem *UCK* (*Ubuntu Customization Kit*) vereinfachen.

Knoppix mit einem anderen Kernel

Soll die eigene Rettungs-CD Treiber für aktuelle SATA-Controller und Netzwerkkarten mitbringen, führt kaum ein Weg am Bau eines eigenen Knoppix-Kernels vorbei. Seit Linux 2.6.16 haben sich viele Schnittstellen des Kernels glücklicherweise kaum verändert, sodass ein Kernel 2.6.24 mit einem ursprünglich mit 2.6.19 gebauten Knoppix wenig Probleme bereitet. Feinarbeit ist lediglich beim HALD (Erkennung von USB-Festplatten), beim Ndiswrapper und seltener bei den Wireless-Tools notwendig. Die Vorgehensweise entspricht weitgehend dem Bauen eines „normalen“ Kernels. Sollten Sie auf einem x86_64-System bauen, müssen Sie in einem i386-Chroot-Käfig arbeiten oder mit einem vorangestellten *setarch i386* vortäuschen, unter i686 zu arbeiten:

- Entpacken Sie den Tarball des Kernels und kopieren Sie die originale Konfiguration des Knoppix (im Live-System unter */boot*) nach *.config* im entpackten Quellcodeverzeichnis.

- Laden Sie die benötigten Patches herunter und spielen Sie diese ein. Squashfs lässt sich beispielsweise als Patch im Kerneltree anbringen:
***patch -p1 < ../squashfs.../.patch*.**
Etwas aufwändiger ist das Union-Dateisystem *aufs*. Sie müssen es zunächst via Subversion auschecken und dann einige Dateien manuell verändern. Im ausgecheckten Verzeichnis finden Sie eine kurze Anleitung.

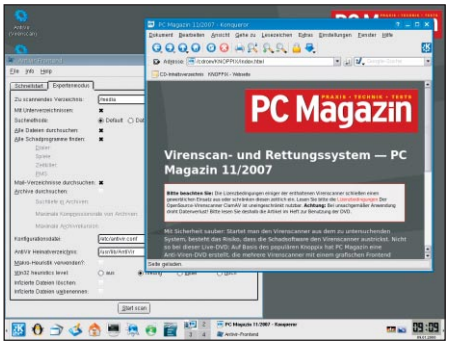
- Erstellen Sie mit *make oldconfig* eine neue Kernelkonfiguration auf Basis der neuen Quellen und der alten Konfiguration und gehen Sie anschließend mit *make menuconfig* die Konfiguration durch. Möglicherweise wollen Sie SATA-Treiber oder Ähnliches statisch einkompilieren oder im alten Kernel noch nicht verfügbare Treiber aktivieren.

- Jetzt bauen Sie den Kernel mit *make*. Erstellen Sie anschließend ein Debian-Paket mit *make deb-pkg*. Dieses können Sie bereits im Chroot-Käfig mit „dpkg -i paket.deb“ installieren.

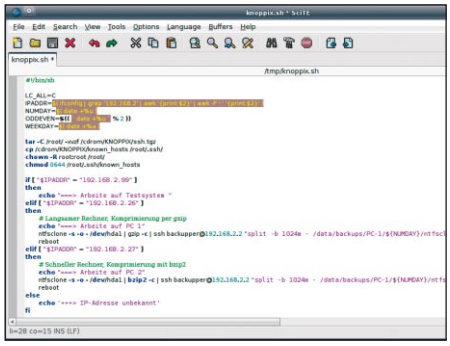
- Cloop wird separat kompiliert. Das von uns getestete *2.622* suchte die Kernelquellen in */usr/src/linux*. Passen Sie den Pfad im Makefile auf das Verzeichnis Ihres Knoppix-Kernels an, um ein passendes Modul zu erhalten.

- Nun müssen Sie die Initrd entpacken, ggf. mounten und die dort enthaltenen Module identifizieren und gegen die Module des frisch kompilierten Kernels austauschen. Beim Original-Knoppix 5.1.1 war zudem eine kleine Änderung am Script *linuxrc* notwendig, damit *aufs.ko* zuerst auf der Initrd gesucht wurde - nur so kann ein alter und ein neuer Kernel parallel genutzt werden.

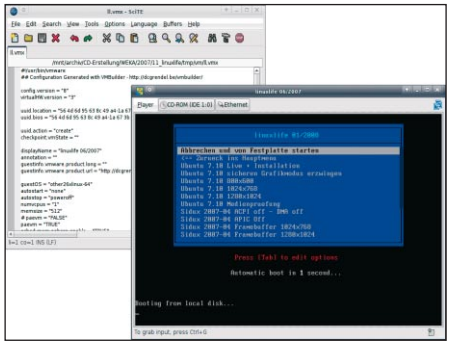
Der neue Kernel - auffindbar unter *arch/i386/boot/bzImage* - und die Initrd kopieren Sie in das Bootverzeichnis (bei Knoppix */boot/isolinux*) und fügen entsprechende Einträge in Ihre *isolinux.cfg* ein.



Typische Knoppix-Modifikation: Die PC-Magazin-Antiviren-CD aus Ausgabe 11/2007 basierte auf Knoppix, das um eigene Erweiterungen ergänzt wurde.



Das Script *knoppix.sh* wird beim Start automatisch ausgeführt - hier reichen wenige Zeilen für eine Knoppix-CD, die eine automatische Datensicherung im lokalen Netz durchführt.



Ideale Testumgebung: Der VMware-Player - Konfigurationsdateien können Sie unter *http://dcgrendel.be/vmbuilder* erzeugen. Alternativ können Sie auch VirtualBox verwenden.

Das Verzeichnis *work* entspricht dem späteren Wurzelverzeichnis der eigenen Knoppix-CD. In ihm können Sie nun zusätzliche Daten unterbringen und im Unterordner *KNOPPIX* das Script *knoppix.sh* anlegen, welches beim Start ausgeführt wird. Sind die Änderungen vorgenommen, erstellen Sie das ISO-Image mit *mkisofs* oder *genisofs*. Ein minimaler Aufruf, der den Bootloader *boot/isolinux/isolinux.bin* erneut integriert, sieht wie folgt aus:

```
mkisofs \
-r -J -pad \
-o meinknoppix.iso \
-b boot/isolinux/isolinux.bin \
-c boot/isolinux/boot.cat \
-no-emul-boot -boot-info-table \
-boot-load-size 4 \
work
```

Da der Bootloader *isolinux.bin* während der Erstellung verändert wird, sollten Sie vor jedem Erzeugen des ISOs eine frische Datei verwenden. Ein unverfälschtes Original finden Sie im *Syslinux*-Tarball. Insbesondere bei größeren Images ist es darüber hinaus ratsam, Sortierinformation heranzuziehen, mit denen Sie dafür sorgen, dass Bootloader und Knoppix-Dateien nahe beieinanderliegen. An erster Stelle steht immer der Bootloader, es folgen die relevanten Dateien und Verzeichnisse, nicht angegebene Dateien werden in alphabetischer Sortierung abgelegt. Bei einer modifizierten Knoppix-CD mit einem Ordner *mp3* könnte die *sort.txt* wie folgt aussehen:

```
work/boot/isolinux/isolinux.bin 200
work/boot/isolinux/isolinux.cfg 190
work/boot/isolinux/german.kbd 180
work/boot/isolinux/menu.c32 170
work/boot/isolinux/ 160
work/boot/ 150
work/KNOPPIX 100
work/mp3 -100
```

Mit dem zusätzlich angehängten Paramter *-s sort.txt* weisen Sie *mkisofs* an, die Sortierung einzubeziehen.

Containern gehen

Zwar kann man mit dem „Quick-and-Dirty"-Remaster viele Anwendungsfälle abdecken, doch spätestens, wenn die eigene Live-CD die 700MB nicht übersteigen soll, bleibt nichts anderes übrig, als Platz für die Nutzlast zu schaffen. Hierfür muss der komprimierte Container entpackt werden.
Knoppix verwendet als Containerformat ein „cloop-komprimiertes" ISO-Image, das ein spezielles Kernelmodul verwendet. Nach dem Download des Tar-Archives wird dieses entpackt und mit einem simplen *make* gebaut, ein *make install* ist nicht notwendig. Anschließend legen Sie den Cloop-Device-Node an:

```
mknod /dev/cloop b 240 0
```

Beim Laden des Moduls ist der Pfad zur Containerdatei anzugeben:

```
insmod cloop.ko file=/pfad/zu/KNOPPIX
```

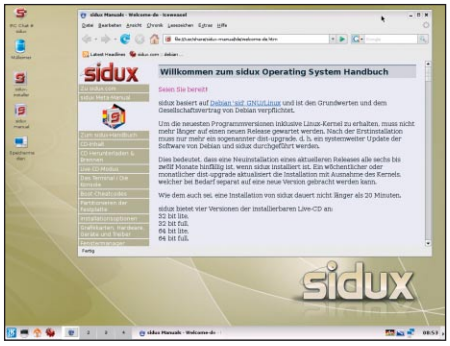
Jetzt kann gemountet werden:

```
mount /dev/cloop /tmp/container
```

Nicht immer lässt sich das Cloop-Modul problemlos laden. Ist das der Fall, können Sie mit

So geht's: Eigene Cheatcodes

Welche Modifikationen für eigene Cheatcodes notwendig sind, hängt davon ab, an welcher Stelle sie den Bootvorgang beeinflussen: Cheatcodes, welche die Suche nach dem Container, dessen Dateisystemtyp oder Lageort beeinflussen, erfordern eine Modifikation der initialen Ramdisk, Cheatcodes die erst danach Dienste starten, können in der Regel als normale Resource-Control-Scripte im */etc/rc5.d* des Containers ausgelesen werden. Ist die Modifikation der Initrd erforderlich, entpacken Sie diese mit *gunzip -c minirt.gz > minirt.img*. Jetzt mounten Sie das Image mit der Option *loop*, hier in den bereits existierenden Ordner *minirt*: *mount -o loop minirt.img minirt/*. Beachten Sie bei der Modifikation von *linuxrc*, dass dieses Script für die „Kompaktshell" *ash* geschrieben wurde - nicht alle von der *bash* bekannten Konstrukte funktionieren. Nach der Anpassung des Scriptes *linuxrc* unmounten Sie das Image und packen Sie es erneut: *umount minirt; gzip -c minirt.img > minirt.gz*. Für größere Änderungen ist es ratsam, ein neues Image zu erstellen und die Daten zu kopieren. So stellen Sie sicher, dass keine unzugewiesenen, aber mit alten Dateifragmenten belegten Blöcke Platz kosten. Das erledigen Sie ganz klassisch mit der Kombination aus *dd* und *mkfs.ext2* oder in einem Schritt mit *genext2fs*. Schlagen die eigenen Cheatcodes fehl, ist es sinnvoll, eine statisch gelinkte *BusyBox* mit auf die Initrd zu packen. Das kostet zwar etwas Platz, macht aber deutlich mehr typische Linux-Befehle verfügbar als in der *ash* und im Verzeichnis */static* der Knoppix-Initrd vorhanden. Werden Kommandos der BusyBox dauerhaft benötigt, empfiehlt es sich, eine abgespeckte Version statisch gegen *uClibc* zu verlinken.



Emanzipierter Abkömmling: *sidux* , das auf *Debian Unstable* basiert, wird derzeit aktiv entwickelt und eignet sich gut für eigene Anpassungen.

Der eigene Kernel erfordert eine eigene Ramdisk - hier bootet ein ansonsten unverändertes Knoppix 5.1.1 mit einem aktualisierten Kernel 2.6.23.

Klassiker: *Isolinux* ist schnell konfiguriert und robust. Mit 640x480 Pixeln großen Hintergründen lässt sich zudem die Optik des Menüs etwas optimieren.

dem Tool *extract_compressed_fs*, welches ebenfalls im Cloop-Paket kompiliert wird, ein unkomprimiertes ISO9660-Dateisystem erzeugen und dann dieses mit dem im Kernel enthaltenen Loopback-Treiber mounten. Da es sich auch beim Container um ein read-only-Dateisystem handelt, muss der Inhalt erst kopiert werden:

```
rsync -av /tmp/container/ build/
```

Um im Verzeichnis *build* arbeiten zu können, sollten Sie einige spezielle Dateisysteme mounten und */tmp* sowie */root* des umgebenden Wirtes verfügbar machen:

```
mount -t proc none build/proc
mount -o bind /dev build/dev
mount -o bind /tmp build/tmp
mount -o bind /root build/root
```

Jetzt folgt der Wechsel in den Chroot-Käfig:

```
LC_ALL=C chroot build
```

Im Chroot können Sie sich nach Belieben austoben, Software mit *apt-get remove* löschen (OpenOffice.org ist ein Kandidat, der besonders viel Platz schafft) oder zuvor nach */tmp* heruntergeladene Pakete mit *dpkg -i* installieren. Von einem kompletten *apt-get upgrade* raten wir indes ab: Knoppix verwendet eine Mischung aus Debian Stable, Testing, Backports und Knoppers eigenen Paketen, die zum Release-Zeitpunkt einer Knoppix-Version immer gut abgestimmt ist, aber mit Sicherheit eine Aktualisierung aller Programme nicht gut verträgt. Sind die Modifikationen beendet, steigen Sie mit *exit* aus dem Chroot aus und unmounten die Dateisysteme:

```
umount build/root
umount build/tmp
umount build/dev
umount build/proc
```

Vergewissern Sie sich mit *cat /proc/mounts*, dass wirklich nichts mehr eingehängt ist. Ein versehentlich im Container integriertes */proc* kann einige Hundert Megabyte okkupieren. Jetzt können Sie den Knoppix-Container neu aufbauen. Hierfür kommt wieder *mkisofs* zum Einsatz, das resultierende Dateisystem wird jedoch in einer Pipeline komprimiert:

```
mkisofs -R -U \
-V „KNOPPIX-Container“ \
-hide-rr-moved \
-cache-inodes  \
-pad build | \
create_compressed_fs \
-f /tmp/KNOPPIX.tmp \
-B 65536 - KNOPPIX
```

Ältere Versionen von *create_compressed_fs* benötigen keine Angabe einer temporären Datei (hier mit dem Parameter *-f*), das „-“ kennzeichnet die Standardeingabe, als letztes Argument folgt der Name der erzeugten Containerdatei, hier *KNOPPIX* im aktuellen Verzeichnis. Sollten Sie *create_compressed_fs* nicht installiert haben, kopieren Sie einfach das Binary aus dem Cloop-Arbeitsverzeichnis nach */usr/bin* oder */usr/local/bin*.
Mit dem neuen Container können Sie jetzt wie oben beschrieben ein neues ISO-Image erstellen, das Sie dann in einer virtuellen Umgebung wie VMware oder VirtualBox testen können.
Enthält das eigene Knoppix basierte Linux ausschließlich freie Software und Programme, spricht auch nichts gegen den Weitervertrieb. Bieten Sie die Quellcodes der LGPL- und GPL-lizenzierten Programme an, entfernen Sie „Knoppix“ aus dem Splashscreen und die *index.html* - und vertreiben Sie das Produkt nicht als *Knoppix*. **jkn**

Bootloader-Konfiguration

Quasi-Standard bei Live- und Multiboot-DVDs ist der Bootloader *Isolinux* aus der Syslinux-Familie. Wir raten zur Vanilla-Version von *kernel.org*.
Konfiguriert wird Isolinux in der Datei */boot/isolinux/isolinux.cfg*. Die klassische Knoppix-Konfiguration lädt einen Splashscreen und zeigt dann einen Bootprompt an. Mit Enter wird dann die als Default gekennzeichnete Konfiguration geladen. Dieses Schema ist ideal, wenn in 95% der Fälle die Standardkonfiguration geladen wird, stört jedoch, wenn mehrere Einstellungen gleichberechtigt existieren. Wir empfehlen deshalb eine Konfiguration, bei der via *isolinux.cfg* die Menükomponente *menu.c32* geladen wird, welche ein einfaches Menü anzeigt.
Mittels *vesamenu.c32* können auch PNG-Grafiken als Hintergrundbild geladen werden - deutlich komfortabler als die *klassischen* Splashes im *LSS*-Format, die mit einem beiliegenden Perl-Script erzeugt werden müssen. Werden Submenüs benötigt, erreichen Sie dies einfach, indem Sie via *menu.c32* oder *vesamenu.c32* eine weitere Konfigurationsdatei laden, ein vollständiges Konfigurationsbeispiel finden Sie auf der Heft-DVD:

```
LABEL sub
MENU LABEL => Expertenmenue
KERNEL vesamenu.c32
APPEND expert.cfg
```