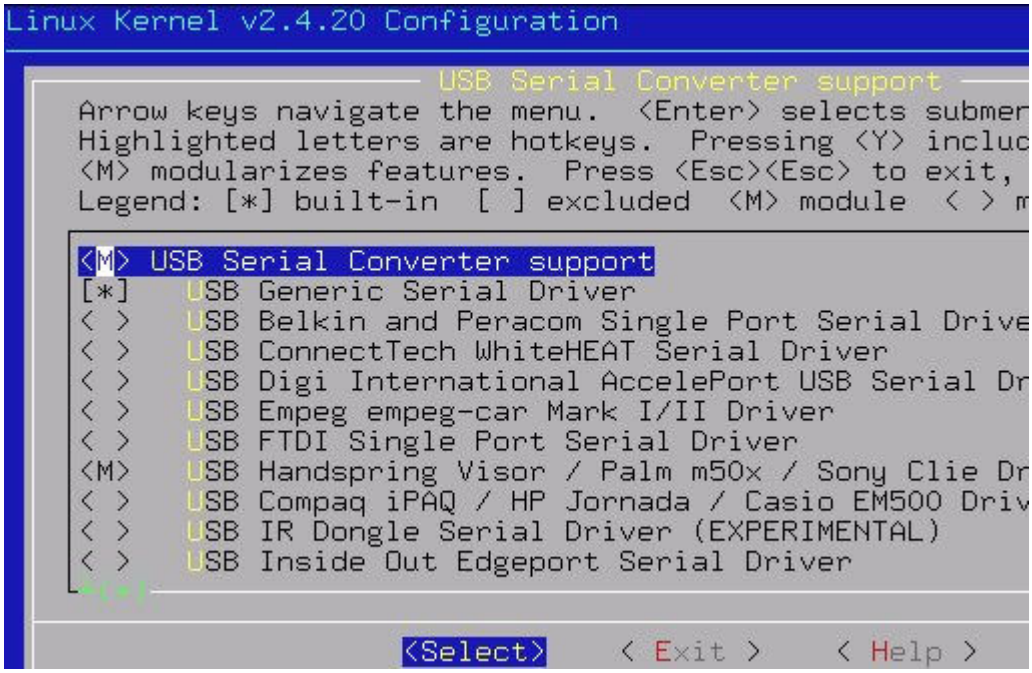
Anleitung - Linux-Kernel kompilieren und installieren

Manchmal ist es erforderlich, einen neuen bzw. anderen Linux-Kernel zu kompilieren und zu installieren. Sei es, weil die Entwickler einen neuen Kernel veröffentlicht haben oder vielleicht die eine oder andere Hardware nur mit einer bestimmten Kernel-Version funktioniert. Aber auch wenn man nur Änderungen an dem Kernel vornimmt, muss man den Kernel neu kompilieren. Das Kompilieren stellt an sich gar keine große Schwierigkeiten dar, wenn man einmal die Routine und den Vorgang begriffen hat.



Der Kernel hält eine riesige Anzahl an Treibern bereit. Man muss nun die passenden Treiber aktivieren. Dies macht man entweder als Modul oder fest in den Kernel.

Zuerst muss man sich natürlich den geeigneten Kernel von www.kernel.org herunterladen. Hier sei gesagt, dass man von den ungeraden Kernel-Versionen die Finger lassen sollte, denn dies sind Entwickler-Versionen und es nicht garantiert, dass dieser Kernel stabil läuft. Ungerade bedeutet, dass die zweite Ziffer eine ungerade Zahl ist, also 2.3.xx oder 2.5.xx.

Grundlegendes: Welche Hardware habe ich? Als Modul oder fest einkompilieren?

Bevor in diesem Artikel auf das Einrichten und Kompilieren des Kernels eingegangen wird, sollen noch ein paar grundlegende Sachen zum Thema Kernel geklärt sein, denn gerade für den Linux-Einsteiger ist die ganze Thematik doch ein wenig abstrakt. Einer der wichtigsten Aufgaben des Kernels ist es, Treiber für die Hardware des Computers bereit zustellen. Da der Computer nicht nur aus einer Grafikkarte und Prozessor besteht, muss man sich schon ein wenig mehr mit der in seinem Computer verbauten Hardware beschäftigen, denn im Kernel muss man angeben, welcher Chipsatz auf dem Mainboard verbaut, welche Soundkarte man besitzt, ob die Festplatte eine IDE oder SATA-Schnittstelle hat, usw.

Da man detailierte Kenntnisse von seiner Hardware haben muss, stellt sich das Kompilieren und Einrichten des Kernels oftmals als sehr schwierige Angelegenheit dar. Doch mit ein paar Tricks kann auch diese Hürde leicht überprungen werden und man erhält einen optimal auf das System eingestellten und stabilen Kernel.

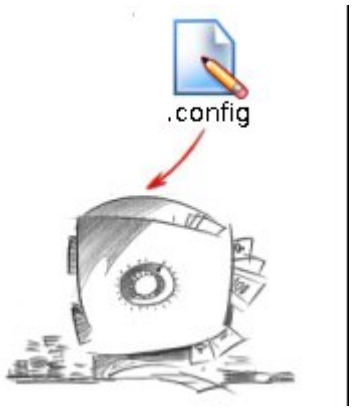
Eine riesige Hilfe beim Herausfinden der verbauten Hardware stellen so genannte Live-CD's dar. Jeder hat bestimmt schon einmal etwas von den CD's gehört, von denen man ein komplettes Linux-System booten kann, ohne Linux auf dem Rechner vorher installiert zu haben. Bekannte Live-CD's sind Knoppix, die Gentoo- oder Suse-Live-CD. Diese Live-CD's erkennen die Hardware automatisch und nachdem man das Linux-Betriebssystem komplett geladen hat, kann man sich mit dem Befehl "cat" im Verzeichnis /proc/ sämtliche Informationen zu der Hardware im Rechner anzeigen lassen. Gebt einfach mal "cat /proc/" gefolgt von einem doppeltem TAB in die Konsole ein und es erscheint eine Menge an Möglichkeiten, sich Informatuionen anzeigen zu lassen.

Beispiel: PCI-Geräte anzeigen lassen

cat /proc/pci

Hat man sich ausgiebig über die im Rechner verbaute Hardware informiert, sollte man sich nun an das Einrichten des Kernels machen bzw. die entsprechenden Optionen aktivieren. Die nächste Frage kommt dann automatisch: Als Modul (M) oder fest einkompilieren (*)? Doch was bedeutet überhaupt Modul und fest einkompilieren? Dazu ein kurzer Exkurs in frühere Linux-Tage, als es noch keine Module gab. Hier musste man sich vor dem Kompilieren genauestens Gedanken machen, welche Optionen aktiviert werden und welche nicht, denn die Treiber waren fest im Kernel integriert und wurden immer beim Booten geladen. Wie man im späteren Abschnitt dieses Artikels erfährt, wird nach dem Kompilieren der Kernel in Form einer Datei erstellt. Je mehr Optionen man fest in den Kernel eingebunden hat, desto größer ist die Datei und desto mehr wird beim Booten geladen.

Dies hat natürlich den Nachteil, dass auch Treiber geladen werden, die man gerade überhaupt nicht braucht, weil das entsprechende Gerät gar nicht angeschlossen ist (Drucker, Scanner, externe USB-Platte). Außerdem ist man nicht flexibel, beispielsweise wenn man ein neues Gerät anschließt, denn dann muss man den kompletten Kernel neu kompilieren, um den Treiber zu integrieren. Alles in allem ist das feste Einkompilieren nicht sehr praktisch, so dass sich die Linux-Entwickler etwas einfallen lassen haben, nämlich die Module.



Damit man die Einstellungen nicht immer wieder vornehmen muss sollte man die .config-Datei sichern. Darin werden sämtliche Einstellungen gespeichert.

Wenn man einen Treiber als Modul kompiliert, so wird dieser nicht in den Kernel integriert, sondern landet als einzelne Datei im Verzeichnis /etc/modules/. Der Vorteil von Modulen ist zum einen, dass diese nicht automatisch beim Booten geladen werden, sondern der Anwender festlegen kann, was geladen werden soll und was nicht. In einer simplen Textdatei (/etc/modules.autoload/) trägt man nun einfach den Namen des Modules ein, so dass der entsprechende Treiber dann geladen wird. Möchte man aus irgendeinem Grund nicht mehr, dass der Treiber geladen wird, so löscht man den Eintrag wieder. Man kann aber auch während des Betriebs mit dem Befehl "modprobe" ein Modul laden, so dass beispielsweise ein neu angeschlossenes Gerät funktionsbereit ist.

Große Distributionen wie Suse, Mandrake oder RedHat machen sich das Prinzip zu Nutze und kompilieren so gut wie alles als Modul in den Kernel. Wer sich einmal die Kernel-Konfiguration einer

solchen Distribution anschaut, wird wahrscheinlich vor lauter M&M's heißhunger auf die gleichnamigen Schokoladen-Trops bekommen :) Beim Booten und während des Betriebs wird dann regelmäßig überprüft, ob neue Hardware hinzugekommen ist, um dann anschließend das Modul zu laden. Ohne die Module müsste hier jedes Mal ein neuer Kernel kompiliert werden, was sehr zeitaufwendig und auf Dauer sehr nervig wäre. Lediglich wenn man zu einer einer anderen Kernel-Version wechselt, müssen auch sämtliche Module neu kompiliert werden, da diese dann nicht mehr kompatibel sind.

Step By Step

Gehen wir einfach mal davon aus, wir würden den 2.4.23-Kernel kompilieren. Natürlich ist dies nicht mehr der aktuellste Kernel, aber dieses Beispiel funktioniert auch für alle anderen Versionen.Wir laden also die Datei linux-2.4.23.tar.bz2 herunter und speichern diese in dem Verzeichnis /usr/src/ . Dieses Verzeichnis beinhaltet in der Regel die Quelldateien von Linux. Nachdem wir also die gepackte Linux-Datei dorthin gespeichert haben, müssen wir diese noch entpacken.

```
tar xvfj /usr/src/linux-2.4.23.tar.bz2
```

Nachdem dieser Schritt gemacht wurde, kann man nun mit "ls /usr/src/" kontrollieren, ob ein neues Verzeichnis angelegt wurde. Als nächstes müssen wir auf die neuen Linux-Quelldateien zeigen, damit das System weiß, dass in dem neuen Ordner der benötige Quellcode liegt. Das macht man mit folgendem Befehl:

Auf neuen Linux-Kernel zeigen
```
ln -s /usr/src/linux-2.4.23/ /usr/src/linux
```

Kontrolle
```
ls -n /usr/src/
```

Bei der Kontrolle müsste ein Eintrag vorhanden sein, der "... linux -> /usr/src/linux-2.4.23" lautet. Ist dies nicht der Fall, löschen Sie den aktuellen Zeiger mit "rm -r /usr/src/linux" und wiederholen Sie den oben genannten Vorgang.

Linux-Kernel konfigurieren

Der nächste Schritt ist der wohl meistgefürchtete Vorgang eines Linux-Users, denn man benötigt nun sämtliche Informationen zu seinem System. In der Konfigurationsdatei des Kernels müssen Sie nun sämtliche Angaben zu Ihrer Hardware machen und die entsprechenden Komponenten in den Kernel einbinden.

Dazu wechseln Sie in das Verzeichnis "/usr/src/linux" und geben den Befehl "make menuconfig" ein. Damit wir den Kernel später überhaupt kompilieren können, benötigt man auf dem Linux-System . Sollte eine Meldung a la "Befehl unbekannt" oder ähnliches erscheinen, so sind die Pakete "gcc" und "make" nicht auf dem System installiert. Diese muss man noch nachinstallieren. Anschließend erscheint ein textbasierendes Konfigurationsmenü für den Kernel.

In das Linux-Verzeichnis wechseln
```
cd /usr/src/linux
```

Aufruf des Konfigurationsmenü
```
make menuconfig
```

Nun müssen Sie sich durch die vielen Einstellungsmöglichkeiten durchkämpfen und die

entsprechenden Komponenten in den Kernel einbinden. Ist diese Arbeit erledigt, muss man den Kernel kompilieren. Das macht man mit folgendem Befehl:

Linux-Kernel kompilieren
(Kernel-Version 2.4)
make dep bzImage modules modules_install

(Kernel-Version 2.6)
make && make modules_install

Nun können Sie sich kurz entspannen, einen heißen Tee trinken oder vielleicht eine Zigarette anzünden, denn dieser Vorgang dauert ein paar Minuten. Nachdem der Kernel kompiliert wurde, legt das System den Kernel in dem Verzeichnis "/usr/src/linux/arch/i386/boot" die Datei bzImage an. Diese müssen wir nun in das Bootverzeichnis kopieren. Wichtig dabei ist, dass das Bootverzeichnis auch wirklich gemountet ist, ansonsten wird die Datei nicht kopiert! Vorsichtshalber mit "mount /boot/" das Verzeichnis ins Dateisystem hängen.

Kernel ins Bootverzeichnis kopieren
cp /usr/src/linux/arch/i386/boot/bzImage /boot/linux-2.4.23

Hinweis: Der letzte Teil des Befehls (.../boot/linux-2.4.23) sagt, dass die Datei kopiert und unter dem Namen linux-2.4.23 gespeichert wird. Wählen Sie also einen beliebigen Namen aus, den Sie sich merken können. Haben Sie bis hier alle Punkte erfolgreich durchführen können, können Sie stolz von sich behaupten, dass sie einen Linux-Kernel kompiliert haben ;-)

Damit Sie nicht immer wieder die Einstellungen erneut auswählen müssen, sollten Sie auf jeden Fall die Datei /usr/src/linux/.config sichern. In dieser Datei werden die Einstellungen gespeichert. Sollten Sie später einen anderen Kernel benutzen, so brauchen Sie nur diese Datei in das /usr/src/linux/ Verzeichnis kopieren und die Einstellungen überprüfen, ob auch alles so weit stimmt. Selbst wenn Sie eine andere Kernel-Version verwenden möchte, kann man die alte .config-Datei benutzen. So ist es nicht mehr erforderlich, sich durch jeden einzelnen Menüpunkt in der Kernelkonfiguration zu "hangeln".

Bootloader einrichten

All diese Vorgänge bleiben wirkungslos, wenn wir dem System nicht sagen, dass er auch den neuen Kernel booten soll. Dazu müssen wir die Booteinstellungen ändern, was wir in diesem Fall über "LILO" machen. LILO ist ein Bootmanager, über den wir Booteinstellungen vornehmen können. Es gibt aber noch einige andere Bootmanager (Grub, etc). Wenn Sie also über einen anderen Bootmanager verfügen, müssen Sie mal einen Blick in die Readme werfen. Das Prinzip ist aber immer dasselbe.

Zuerst öffnen wir mit einem Texteditor die Konfigurationsdatei von Lilo, nämlich "lilo.conf". Dazu geben wir folgenden Befehl ein:

LILO-Konfigurationsdatei
nano -w /etc/lilo.conf

Daraufhin sehen wir die Bootroutine, die wir nun ändern müssen. Um auf der sicheren Seite zu sein, legen wir den neu kompilierten Kernel als zusätzliche Bootmöglichkeit an. Sollte der neue Kernel fehlerhaft sein, kann man dann noch von dem altem, funktionierendem Kernel starten. Hier ein Ausschnitt aus der lilo.conf:

```
image = /boot/bzImage   #alter Kernel
```

```
        root = /dev/hda1
        label = linux
        read-only

image = /boot/linux-2.4.23   #neuer Kernel
        root = /dev/hda1
        label = linux
        read-only
```

Wie man gut erkennen kann, haben wir zu dem alten Kernel noch eine alternative Bootmöglichkeit abgegeben. Dazu sollten Sie nach "image =" den Pfad zum neuen Kernel angeben. Da wir im vorigen Schritt den Kernel nach /boot kopiert und ihn linux-2.4.23 genannt haben, müssen wir diesen Pfad nun auch in der lilo.conf angeben. Beim Neustart erscheint dann das Auswahlmenü, wo Sie entscheiden können, ob Sie von dem neuen Kernel oder vom alten Kernel starten möchten.

Die Änderungen in der lilo.conf müssen nun noch gespeichert werden. Dies macht man mithilfe folgendem Befehl:

Speichern von lilo
/sbin/lilo

Nun können Sie Ihr System neustarten und die vom neuen Kernel booten!

Schlussworte

Lassen Sie sich nicht von dem komplexen Vorgehen entmutigen. Hat man einmal einen Kernel erfolgreich kompiliert und installiert, stellen die nächsten Änderungen am Kernel für Sie keine Probleme mehr da.

Diese Anleitung müssen Sie nicht nur beim installieren eines komplett neuen Kernels benutzen, sondern auch, wenn Sie Änderungen a, aktuellen Kernel vornehmen.

Zurück zur Startseite