Der Linux-Kernel

Autor: Erwin Dogs (*edogs@t-online.de*)
Layout: Matthias Hagedorn (*matthias.hagedorn@selflinux.org*)

Grundsätzliche Arbeitsweise des Linux-Systems

Inhaltsverzeichnis

1 Der Kernel

Der Betriebssystemkern (Kernel) erledigt in einem Betriebssystem die grundlegendsten Aufgaben. Dazu gehören: Verwaltung von Prozessen, Verwaltung des Dateisystems, Steuerung von Hardwareressourcen, Speicherverwaltung. Der Kernel ist somit das Herz des Betriebssystems. Unter Linux beinhaltet der Kernel auch Gerätetreiber, die das System braucht (als Runtime-Module oder fest eingebaut). Nichts funktioniert ohne einen Kernel, und genau das ist es, was Linux zu Linux macht, denn wenn wir von Linux sprechen, meinen wir eigentlich den Kernel in der aktuellen stabilen `Version 2.4.x`.

Aber ein Kernel allein macht noch kein Betriebssystem aus, sondern muss auch irgendwie vom Benutzer angesprochen werden können, damit es überhaupt Arbeit gibt, die er verrichten kann.

Der Kernel liegt als Datei (`bzImage`) auf einem Datenträger vor, und muss für den Start in den Hauptspeicher eingelesen werden.

Dazu gibt es verschiedene Methoden, welche sich dadurch unterscheiden, wo die Kerneldatei liegt. Wir kennen 3 verschiedene Orte, an denen ein bootfähiger Kernel residieren kann:

1. Internet
2. Intranet
3. Localhost - Diskette, Festplatte, CD-ROM, EPROM, EEPROM o.ä.

Betrachtet wird hier die Variante Diskette, welche fast zwingend neben der Betriebsboot-Möglichkeit am **localhost** vorhanden sein sollte.

Praxisübung:

> Suchen Sie die Kerneldatei, mit der Ihre Distribution den Rechner zum Booten gebracht hat. Kopieren Sie diese Datei im `raw-Modus` auf eine roh (ohne Erstellung eines Filesystems) formatierte 3,5"-Diskette:

```
root@linux / # dd if=KERNELFILE of=/dev/fd0u1440
```

Zur Erprobung booten Sie Ihren Rechner mit dieser Diskette. Dazu ist die Bootreihenfolge im BIOS möglicherweise wie folgt umzustellen:

```
A: CDROM: C:
```

Dieser Bootvorgang wird mit grosser Wahrscheinlichkeit fehlschlagen, und das ist gut so. Der Kernel der Distribution hat die Root-Partition in sich gespeichert. Diese stimmt in den seltensten Fällen mit der tatsächlichen Root-Partition überein. Abhilfe schafft eine Behandlung der Diskette mit dem Befehl `rdev`.

Falls das Booten nicht fehlgeschlagen ist, weil die Root-Partition stimmt, ist es lohnenswert, einmal mit `rdev` eine falsche Root-Partition in den Kernel zu schreiben, um einen fehlerhaften Bootvorgang (Kernel bleibt stehen) beobachten (und später erkennen) zu können.

Ein auf dem eigenen Rechner selbst compilierter Kernel hat die richtige Root-Partition in sich gespeichert.

2 Init

Als **"Kommunikationsschnittstelle"** zwischen dem Kernel und dem Benutzer fungiert der `init`-Prozess.

`init` gibt Aufgaben an den Kernel weiter und empfängt Anweisungen vom Kernel (etwa "init starte bitte Programm xy als Prozess yz"). Das ist die Aufgabe von `init`. Außerdem überwacht `init`, dass einige bestimmte Programme, die

das System zum Arbeiten benötigt, immer laufen und startet diese neu, wenn diese Prozesse einmal beendet werden.

Beim Systemstart und -herunterfahren startet bzw. stoppt `init` die benötigten Programme.

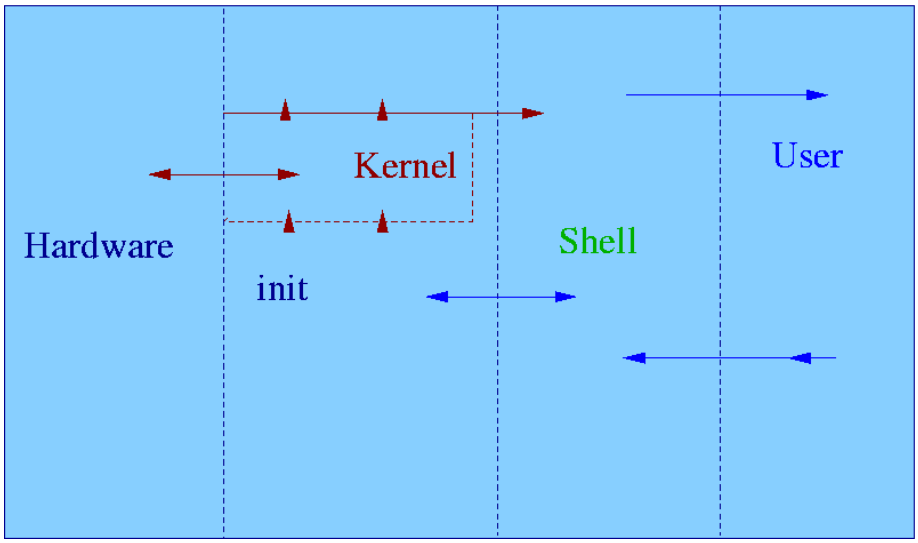
Die Informationen, die der `init`-Prozess braucht, um zu wissen, welche Programme gestartet werden müssen, findet er in der Regel im Verzeichnis /etc/init.d. In diesem Verzeichnis liegen die Startskripte für die meisten Dienste. Diese Skripte enthalten Anweisungen, wie welche Programme gestartet und auch gestoppt werden. Nun wird in sogennannten **"Runlevels"** festgelegt, welche Sammlung von Diensten in einem bestimmten Systemmodus gestartet wird. Dazu liegen in den entsprechenden Runlevelverzeichnissen (i.d.R. /etc/rc(s,0-6).d) Links auf die Skripte in /etc/init.d. Diese Links können zum Starten als `Sxx-Skript` (S="start") oder zum Stoppen als `Kxx-Skript` (K="kill") aufgerufen werden. XX steht für eine Zahl, z.B. `S02checkfs`. Mit dieser kann die Priorität und damit Reihenfolge des Aufrufs festgelegt werden.

Die Information, in welches Runlevel das System standardmäßig startet, in welchem Verzeichnis die Links liegen und auch noch einige andere, finden sich in der Konfigurationsdatei `/etc/inittab`.

Aber nicht nur beim Systemstart, sondern auch im laufenden Betrieb verrichtet `init` ständig seine Arbeit. Dazu muss es eine Umgebung geben, in der der Benutzer Befehle an das System geben kann. Dies ist unter Linux die Shell.

3 Shell

Die **Shell** ist die Schnittstelle zwischen Benutzer und System. In ihr gibt der Benutzer Befehle an das System und die **Shell** führt diesen Befehl aus, indem sie `init` veranlasst, das ausführbare Programm, das hinter diesem Befehl steht, zu starten. Die Standardshell unter Linux ist die **bash** (Bourne Again Shell), die modernste und mächtigste aller Shells. Prinzipiell jedoch kann jede Unix-Shell (z.B. Bourne-Shell, C-Shell, Korn-Shell...) unter Linux verwendet werden.



kernel2.png

Der Weg eines Befehls, den ein User eingibt, wird schematisch im Schaubild oben dargestellt:

User->Shell->init->Kernel->Hardware->Kernel->init->Shell->User

4 Erzeugen eines eigenen Kernels

4.1 Voraussetzungen

Um einen eigenen Kernel zu übersetzen, brauchen Sie einen C-Compiler, der die Quelltexte ins Binärformat übersetzt. Ob der C-Compiler installiert ist, erfahren Sie mit dem folgenden Befehl:

```
root@linux / # gcc --version
```

oder

```
root@linux / # cc --version
```

Nach Eingabe dieses Befehls sollte die Versionsnummer auf dem Schirm erscheinen.

4.2 Holen der Kernel-Quellen

Nun brauchen Sie noch die aktuellen Kernel-Quellen. Die Quellen des unveränderten Original-Kernels finden Sie auf

http://www.kernel.org

Schauen Sie auch auf der Downloadseite ihres Distributors, der oft eigene (angepaßte) Versionen des Kernels im `RPM`- oder `DEB`-Format bereitstellt.

Die Quelltexte des Kernels sollten Sie nach /usr/src/linux entpacken (bei der Installation eines Paketes wird das oft bereits erledigt).

Wechseln Sie in das /usr/src Verzeichnis mit `cd /usr/src`, um zu schauen, ob dieses Verzeichnis existiert. Wenn ja, kann es auch schon losgehen.

Wenn Sie die aktuelle Version des Originalkernels von http://www.kernel.org heruntergeladen haben, kopieren Sie die Datei `linux-2.x.xx.tar.bz2` in das Verzeichnis `/usr/src` und entpacken Sie das Archiv mit dem folgenden Befehl:

```
root@linux /usr/src/ # tar xvfj linux-2.x.xx.tar.bz2
```

Nach dem Entpacken ist das Verzeichnis `/usr/src/linux` entstanden.

4.3 Hinweis zur Versionsnummer des Kernels

Es werden im Grunde zwei Arten von Kernels unterschieden. Das sind zum einen die **stabilen Kernel**, zum anderen die **Entwickler-Kernel**, an denen aktuell gearbeitet wird und die nicht stabil genug für die Benutzung sind. Diese unterscheiden sich anhand der Versionsnummer. Eine gerade Zahl an der zweiten Stelle der Versionsnummer kennzeichnet den stabilen Kernel-Zweig, eine ungerade den aktuellen Entwickler-Kernel. Also sind etwa 2.0, 2.2 und 2.4 **stabile Kernel**. Die Versionen 2.1, 2.3 und 2.5 dagegen sind **Entwickler-Kernel**.

Version 2.5 wird irgendwann mit den Änderungen und Erweiterung eingefroren, das heißt, es werden keine neuen Features hinzugefügt und getestet. Wenn die Testphase abgeschlossen ist, wird der Kernel als stabil gekennzeichnet und erhält die Versionsnummer 2.6 (oder möglicherweise auch 3.0).

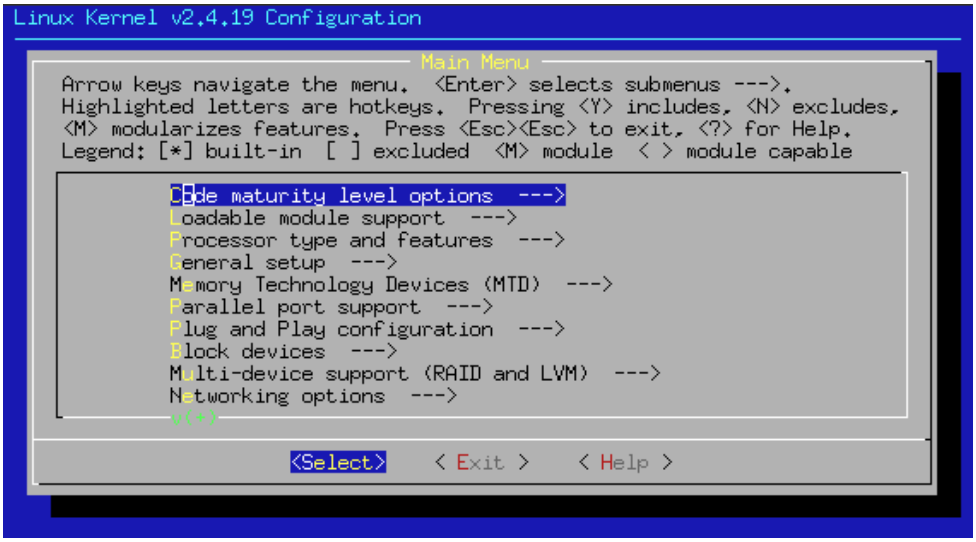
Die Zahl, die darauf folgt, ist der Patchlevel des Kernels. Mit jedem Patchlevel sind Korrekturen in den stabilen Kernel

eingeflossen und geben somit Auskunft über die Aktualität des Kernels.

4.4 Konfiguration des Kernels

Sie haben im wesentlichen drei Möglichkeiten, einen neuen Kernel zu konfigurieren.

* `make config` ist die spartanischste von allen: innerhalb des Konsolenfensters werden dem Benutzer Fragen gestellt, die er einzeln beantworten muss. Da dies mit der Zeit sehr umfangreich und unübersichtlich geworden ist, ist diese Methode nicht zu empfehlen.
* `make menuconfig` stellt ein grafisches Menü in einer Konsole bereit. In diesem Menü kann man Konfigurationsoptionen auswählen. Meistens werden die Möglichkeiten [X] [ ] [M] (aktiviert, deaktiviert und als Kernelmodul kompiliert) angeboten. Um diesen Weg zu nutzen, benötigt Ihr System die **ncurses**-Bibliothek, die jedoch von den meisten Distributionen von ganz allein installiert wird.
* `make xconfig` bietet ebenfalls ein Auswahlmenü an, aber unter der XWindow-Oberfläche. Hierzu müssen Sie **Tcl/Tk** installiert haben.



make menuconfig in Aktion

Schauen Sie sich zunächst genau die Auswahloptionen an und vergleichen Sie, welche Hardware Sie in ihrem Computer haben (eventuell schauen Sie auch einmal in die Ausgaben von `lspci` und `dmesg`). Schauen Sie ggf. auf der Homepage des Herstellers nach, ob Sie eine Dokumentation zu Ihrer Hardware finden.

Nachdem Sie die Auswahlentscheidungen getroffen haben, können Sie Ihre Konfiguration speichern. Die Konfiguration wird dann in die Datei

`/usr/src/linux/.config`

geschrieben.

4.5 Übersetzung des Kernels

Nach der Konfiguration erfolgt die Übersetzung mit den Befehlen:

```
root@linux / # make dep
root@linux / # make clean
```

```
root@linux / # make bzImage
root@linux / # make modules
root@linux / # make modules_install
```

4.6 Installation des Kernels

Nach dem Kompilieren (die Kompilierungszeit ist abhängig von der Rechnenleistung Ihres Computers und kann mehrere Stunden oder auch nur einige Minuten dauern) kopieren Sie den Kernel in Ihre Boot-Partition bzw. Ihr `/boot`-Verzeichnis. Sichern Sie bitte vorher den funktionierenden Kernel! Eine Befehlsfolge könnte etwa wie folgt aussehen:

```
root@linux / # cd /usr/src/linux
root@linux / # cp /boot/vmlinuz /boot/vmlinuz.old
root@linux / # cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz
root@linux / # cp /boot/System.map /boot/System.map.old
root@linux / # cp System.map /boot
```

Um den neuen Kernel booten zu können, müssen Sie dem Bootloader noch mitteilen, wo er den neuen Kernel findet.

Öffnen Sie bei Verwendung von LILO die Datei /etc/lilo.conf mit einem Texteditor und fügen Sie Einträge der Form

/etc/lilo.conf

```
...
# Neuer Kernel:
image = /boot/vmlinuz
label = kernel_new

# Backup-Kernel:
image = /boot/vmlinuz.old
label = kernel_old
...
```

hinzu.

Nun führen Sie noch als `root /sbin/lilo` aus.

4.7 Starten des neuen Kernel

Nun können Sie ihren neuen Kernel booten. Geben Sie dazu als root

```
root@linux / # reboot
```

oder

```
root@linux / # shutdown -r now
```

ein.

Beim Erscheinen des Bootmenüs wählen Sie den Eintrag mit dem neuen Kernel aus. Damit bootet der Computer mit Ihrem neuen Kernel.

Sollte der Computer aus irgendwelchen Gründen nicht booten, so schauen Sie sich die Fehlermeldungen genau an, sie geben Aufschluß über die Ursachen. Resetten Sie in diesem Fall Ihren Computer und booten Sie ihn erneut mit dem alten Kernel. Sie finden die Fehlermeldungen, falls sie nicht ohnehin auf der Konsole ausgegeben wurden, in der Datei `/var/log/messages`.

Weitere Infos zum Erstellen eines eigenen Kernels (in englischer Sprache) finden Sie auch hier:

http://www.tldp.org/HOWTO/Kernel-HOWTO.html