Inhaltsverzeichnis

1 Neues im Kernel 2.6

1.1 Konfiguration & Build System

Das Konfigurationssystem wurde neu geschrieben, von der Bedienung her hat sich nichts geändert, jedoch wurde für `make xconfig` das Xlib-Interface durch eine Qt-Oberfläche ersetzt. Alternativ ist nach großem Protest auch ein Gtk+-Interface (`make gconfig`) hinzugekommen.
Auch am Build System wurde gearbeitet, jedoch nicht die angefangene Neuimplementierung übernommen, sondern das bestehende System verbessert. Jetzt fällt der „`make dep`“-Schritt weg und die Trennung zwischen Kernel- und Modul-Kompilierung wurde aufgehoben.

1.2 Module

Das Modulformat wurde komplett geändert, daher ist ein Ersatz der modutils nötig, module-init-tools genannt. Diese ersetzen zwar die modutils, erstellen aber Kopien (z.B. modprobe.old), die aufgerufen werden, wenn ein Kernel $< 2.5.X$ (FIXME, ca. .48) gestartet wurde.
Bei der Installation aus den sollte man unbedingt die README-Datei lesen und sich auch danach richten, da man bei falscher Installation auch die Modutils für der 2.4er Kernel überschreiben kann.

1.3 neue Treiber

Viele Treiber wurden neu geschrieben, einige neue sind hinzugekommen, die folgende Liste ist alles andere als komplett:

- ALSA[1] ersetzt praktisch OSS[2]
- lm_sensors, das Paket zur Überwachung von Hardware (z.B. Temperatur) ist jetzt fester Bestandteil im Kernel
- User Mode Linux
- Support für neue Architekturen: x86-64, ppc64
- 32-bit `dev_t` (12 bit major, 20 bit minor)
- SCTP (Streaming Control Transmission Protocol)
- Serial ATA
- Crypto API
- DVB layer
- Hotplug CPU support

[1] **A**dvanced **L**inux **S**ound **A**rchitecture
[2] **O**pen **S**ound **S**ystem

2 Wie baue ich mir einen eigenen Kernel

2.1 Warum überhaupt?

Vorteile eines speziellen, genau auf den jeweiligen Computer abgestimmten Kernels bestehen in der Schnelligkeit, die durch Compileroptimierungen und die minimale Auswahl an Treibern erreicht wird, sowie dem geringeren Platzbedarf und in den meisten Fällen auch der erhöhten Stabilität, auch aus Gründen der Minimalisierung.

2.2 Voraussetzungen & benötigte Programme

Hier sollte man vor allem die Dateien `$KERNELSRC/README`[3] und `$KERNELSRC/Documentation/Changes` zu Rate ziehen, die mehr oder weniger regelmäßig aktualisiert werden.
Zur Erstellung des Kernels werden benötigt[4]:

- GNU Make 3.78
- GCC 2.95.3
- NCurses für menuconfig
- Qt für xconfig
- Gtk+ für gconfig

Außerdem werden natürlich die Kernelquellen benötigt. Zuerst sollte man sich das Quellenpaket organisieren, z.B. durch:

```
MIRROR=ftp.de.kernel.org
wget ftp://$MIRROR/pub/linux/kernel/v2.6/linux-2.6.0-test6.tar.bz2
```

Dann entpackt man dieses Paket an geeigneter Stelle:

```
cd ~/src
bunzip -c linux-2.6.0-test6.tar.bz2 | tar xv
```

Im Folgenden wird mit Kernelquellen oder Quellenbaum auf das Verzeichnis `~/src/linux` verwiesen.
Damit der frisch gebackene Kernel auch läuft, sollte folgendes vorhanden sein:

- binutils 2.12
- module-init-tools 0.9.9
- util-linux 2.10o

[3] `$KERNELSRC` bezeichnet die Stelle, an der der Kernelquellenbaum liegt, üblicherweise `/usr/src/linux`

[4] neuere Versionen, soweit vorhanden, sind natürlich zu bevorzugen, Ausnahmen werden erwähnt

2.3 Konfiguration

Als Ziel der Konfigurationsphase entsteht eine Datei namens `.config`, die alle Konfigurationsoptionen enthält. Theoretisch ist es möglich, diese Datei per Hand zu erstellen bzw. zu editieren, das wird jedoch aufgrund der Komplexität und der zum Teil vorhandenen Abhängigkeiten nicht empfohlen.
Die meisten Konfigurationsoptionen bestehen aus „Multiple Choice"-Fragen, oft kann man zwischen *nein (no,n)*, *ja (yes,y)* und *modular (m)* wählen, zum Teil etwas aus einer Liste auswählen, und selten muss man einen numerischen Wert oder eine Zeichenkette direkt eingeben.

`make help`: Bietet eine kleine Übersicht über vorhandene Konfigurations- und Erstellungsmöglichkeiten.
`make config`: Für alle Optionen muss man sich zwischen den vorgegebenen Möglichkeiten entscheiden.
`make oldconfig`: Eine vorhandene `.config` wird gelesen und es werden nur Fragen zu Optionen gestellt, die neu hinzugekommen sind. Dies geschieht üblicherweise, nachdem der Kernel gepatcht wurde.
`make menuconfig`: Mittels einer menügesteuerten Textoberfläche sollte man alle Optionen konfigurieren.
`make xconfig`: Eine graphische Konfigurationsoberfläche aufbauend auf dem Qt-Toolkit wird erstellt und gestartet.
`make gconfig`: Eine Gtk+-basierte Oberfläche wird zur Konfiguration genutzt.

Eher von Bedeutung für Entwickler und Distributoren sind `make defconfig`, `make allmodconfig`, `make allyesconfig` und `make allnoconfig`

2.4 Kompilierung & Installation

Die Kompilierung lässt sich jetzt im Normalfall mit einem Befehl abhandeln: `make`. Dies baut den Linuxkernel (`vmlinux`, bzw. `bzImage` als Standard auf x86) sowie alle konfigurierten Kernelmodule. Dieser Schritt dauert je nach Rechner so zwischen 2 Minuten und 1 Tag.
Neu ist, dass weniger vom eigentlichen Kompilierungsprozess zu sehen ist. Zu jedem erstellten Objekt wird nur eine Zeile ausgegeben, die ungefähr erahnen lässt, was gerade geschieht. Fehler und Warnungen werden selbstverständlich weiterhin an den Nutzer weitergegeben. Möchte man genau wissen, was passiert, so startet man den Build-Prozess mit „`make V=1`".
Während der erledigte Teil auch mit normalen Benutzerrechten zu bewerkstelligen, sind im Folgenden Root-Rechte erforderlich.
Ist der Kompilierungsprozess erfolgreich abgeschlossen, kopieren wir den Kernel an eine geeignete Stelle:

```
cp arch/i386/boot/bzImage /boot/Linux-2.6.0-test6-1
```

Danach müssen noch die Module installiert werden:

```
make modules_install
```

Das war es prinzipiell schon, müsste nicht noch unser präferierter Bootmanager von dem neuen Kernel erfahren. Dazu editieren wir entweder `/etc/lilo.conf` oder `/boot/grub/menu.lst`, jenachdem, ob Lilo oder Grub benutzt wird. Im Falle von Lilo noch einmal `lilo` starten, um den Bootsektor neue zu schreiben und der neue Kernel ist installiert.
Um den neuen Kernel zu testen, ist es leider unumganglich, die schöne Uptime kaputtzumachen und der Computer neu zu starten. Im Menü des Bootmanagers (ja, das ist heute üblich, es gehrt natürlich auch ohne) wählt man seinen frisch gebackenen Kernel aus und hofft, dass man bei der Konfiguration alles richtig gemacht hat, sonst ist das Vergnügen von kurzer Dauer.

2.5 Änderungen gegenüber 2.4.x

- `make dep` überflüssig
- `make bzImage` und `make modules` zusammengefasst