Kernelgenerierung

Adrian Reber

adrian@fht-esslingen.de
Hochschule für Technik - Fachhochschule Esslingen

http://lisas.de/~adrian/kernel/
24. Juni 2001

Vorwort

Dieser Text gibt eine Anleitung zu Konfiguration, Kompilation und Installation des Linux-Kernels für ix86-basierte Systeme.

Alle hier gegebenen Informationen basieren auf der aktuellen Kernelserie der Version 2.4.x. Aktueller Kernel zum Zeitpunkt der Verfassung dieses Textes ist 2.4.3

Entstanden ist dieses Dokument im Rahmen der "Vorlesung Betriebssysteme 2" bei Professor Dr. rer. nat. Heinrich Weber.

Inhaltsverzeichnis

Kapitel 1

Grundlagen

1.1 Vorbemerkungen

1.1.1 Allgemein

- Alle Pfade, wenn nicht absolut angegeben, beziehen sich auf das Verzeichnis, das die Sourcen des Linux Kernels enthält. Normalerweise ist das /usr/src/linux. Dies ist aber nicht zwingend.

1.1.2 Versionsnummern

Einen aktuellen Linux Kernel gibt es immer in zwei Versionen. Eine stabile (stable) Version und eine Entwicklerversion (development). Die stabile Version hat eine gerade Versionsnummer (2.4.x) und die Entwicklerversion hat eine ungerade Nummer (2.5.x). Der erste stabile Kernel war 1.0.x, gefolgt von 1.2.x und 2.0.x bis zur aktuellen Version 2.4.x. Die Entwicklerkernel waren immer dazwischen mit den Versionsnummern 1.1.x, 1.3.x, 2.1.x und 2.3.x. An der Version 2.5.x wird noch nicht aktiv gearbeitet, aber es wird schon stark darüber diskutiert, welche Neuerungen und Änderungen dort gemacht werden sollen.

1.2 Was macht der Kernel überhaupt?

Der Unix-Kernel stellt eine Art Vermittler zwischen den Anwenderprogrammen und der Hardware des Computers dar. Er verwaltet den Arbeitsspeicher des Rechners und sorgt dafür, dass jedes laufende Programm (Prozess) angemessene Anteile der Prozessor-Arbeitszyklen zugewiesen bekommt. Der Kernel stellt eine von der speziellen Hardware unabhängige Schnittstelle zum Zugriff auf diese Hardware zur Verfügung.

Es gibt zwar noch eine Menge weiterer Dinge, für die der Kernel zuständig ist, doch sind dies die wichtigsten, über die jeder Bescheid wissen sollte.

1.3 Warum einen Kernel selbst generieren?

Bei allen aktuellen Distributionen kommt man inzwischen fast nicht mehr in die Situation, in der eine Kernelgenerierung nötig ist. Die Distributoren statten ihre Standardkernels mit eigentlich allem aus, um auch die ausgefallenste Hardware zu unterstützen.

Trotzdem lohnt es sich eigentlich immer, auf einen selbst gebauten Kernel umzusteigen. Die Standardkernel der Distributoren sind mit allen Treibern vollgestopft, so dass sie riesig sind und Funktionalitäten enthalten, die man auf der eigenen Hardware meist nicht in diesem Umfang benötigt. Außerdem kann man den selbst erschaffenen Kernel besser an die eigene Hardware und Bedürfnisse anpassen:

- die richtige Optimierung für die eigene Hardware,
- den richtigen Treiber für den Festplattenkontroller, um auch all die Leistung zu bekommen die die Hardware verspricht,
- usw.
- etc.
- This space is for rent

- How Am I Driving? 1-800-U-HAUL

1.4 Module

Der Linuxkernel bietet die Möglichkeit, Teile, die nicht immer benötigt werden, während der Laufzeit in den Kernel einzubinden und auch wieder zu entfernen wenn sie nicht mehr nötig sind. Diese extreme Flexibiltät hat inzwischen dazu geführt, dass man einen Großteil der Hardwaretreiber als Modul erstellt und dadurch einen noch schlankeren und effizienteren Kernel erhält. Ein weiterer Grund für die Notwendigkeit von Modulen ist die etwas seltsame Hardwarearchitektur auf x86 basierten Systemen. Wenn der Kernel komprimiert größer als 640KB wird, kann er nicht mehr geladen werden.

1.5 Welcher Kernel läuft bei mir?

Um die Versionsnummer des laufenden Linux Kernels auf seinem System zu erfahren benutzt man am besten den Befehl *uname*.

```
> uname -r
2.4.2-ac3
```

1.6 Voraussetzungen

1.6.1 Hardware

Linux läuft nicht nur auf Intel basierten Systemen. Obwohl es auf 386/486 basierten PC-Systemen entwickelt wurde, läuft es inzwischen auf ARMs, DEC Alphas, SUN Sparcs, M68000 (wie Atari und Amiga), MIPS, PowerPC, S390 und anderen Systemen. Einen guten Überblick über die unterstützte Hardware gibt das Verzeichnis linux/arch.

```
/usr/src/cvs/linux/arch > ls -la
total 76
drwxr-xr-x   19 root     root         4096 Apr 14 19:38 .
drwxr-xr-x   17 root     root         4096 Apr 21 12:10 ..
drwxr-xr-x    2 root     root         4096 Apr 19 22:48 CVS
drwxr-xr-x    8 root     root         4096 Apr 19 22:50 alpha
drwxr-xr-x   15 root     root         4096 Apr 19 22:50 arm
drwxr-xr-x    8 root     root         4096 Apr 19 22:50 cris
drwxr-xr-x    8 root     root         4096 Apr 21 11:10 i386
drwxr-xr-x   13 root     root         4096 Apr 19 22:50 ia64
drwxr-xr-x   23 root     root         4096 Apr 19 22:50 m68k
drwxr-xr-x    3 root     root         4096 Apr 14 19:37 math-emu
drwxr-xr-x   19 root     root         4096 Apr 19 22:50 mips
drwxr-xr-x   11 root     root         4096 Apr 19 22:50 mips64
drwxr-xr-x    8 root     root         4096 Apr 19 22:50 parisc
drwxr-xr-x   17 root     root         4096 Apr 19 22:50 ppc
drwxr-xr-x    9 root     root         4096 Apr 19 22:50 s390
drwxr-xr-x    8 root     root         4096 Apr 19 22:50 s390x
drwxr-xr-x    7 root     root         4096 Apr 19 22:50 sh
drwxr-xr-x   10 root     root         4096 Apr 19 22:50 sparc
drwxr-xr-x   10 root     root         4096 Apr 21 11:11 sparc64
/usr/src/cvs/linux/arch >
```

Wie man sieht ist Linux sehr flexibel, wenn es um Hardwareplattformen geht.

1.6.2 Software

Für einen aktuellen Kernel (2.4.x) gibt es bestimmte Voraussetzungen für Software, die nötig ist, um den Kernel zu generieren. Die folgende Tabelle zeigt die wichtigsten Softwarepakete. Natürlich wird davon ausgegangen, dass ein funktionierendes System vorhanden ist.

Softwarepaket	Version	Versionstest
Gnu C	2.91.66	# gcc –version
Gnu make	3.77	# make –version
binutils	2.9.1.0.25	# ld -v
util-linux	2.10o	# fdformat –version
modutils	2.4.2	# insmod -V
e2fsprogs	1.19	# tune2fs
reiserfsprogs	3.x.0j	# reiserfsck 2>&1—grep reiserfsprogs
pcmcia-cs	3.1.21	# cardmgr -V
PPP	2.4.0	# pppd –version
isdn4k-utils	3.1pre1	# isdnctrl 2>&1—grep version

Kapitel 2

Generierung

Die Generierung des Kernels unterteilt sich in mehrere Abschnitte: Download, Entpacken, Konfiguration, Kompilation und Installation.

2.1 Download

Der offizielle Ort für den Linux Kernel ist ftp://ftp.kernel.org. Am besten ist es aber, wenn man sich die Sourcen für den Linux Kernel von einem der zahlreichen Mirror Servern holt. Für Deutschland ist das ftp://ftp.de.kernel.org. Hier an der Hochschule zeigt ftp.de.kernel.org auf einen Rechner (rhlx01.fht-esslingen.de) im Rechenzentrum, so dass man mit einer recht guten Übertragungsrate rechnen kann. Wer hinter einer Firewall sitzt und leider nicht per ftp die Sourcen runterladen kann, der kann auch meist sein Glück per http versuchen.
Der vollständige Pfad zu den Linux Sourcen ist

ftp://ftp.de.kernel.org/pub/linux/kernel/v2.4.

Die Datei LATEST-IS-2.4.x sagt, welche Version die aktuelle ist.
Wer immer auf dem allerneusten Stand bleiben möchte, kann sich natürlich auch per cvs die Sourcen besorgen: `:pserver:cvs@vger.samba.org:/vger`

2.2 Entpacken der Sourcen

Die Kernel Sourcen stehen in mehreren Varianten zur Verfügung:

- komplettes tar Archiv gepackt mit gzip:
  linux-2.4.x.tar.gz
- komplettes tar Archiv gepackt mit bzip2:
  linux-2.4.x.tar.bz2
- als patch gepackt mit gzip:
  patch-2.4.x.gz
- als patch gepackt mit bzip2:
  patch-2.4.x.bz2

Nach dem Runterladen der Sourcen packt man sie üblicherweise unter */usr/src* aus. Beim Entpacken der tar-Archive entsteht dabei ein neues Verzeichnis mit dem Namen *linux*, welches nun all die Dateien enthält, die nötig sind um einen Kernel zu übersetzen. Am besten ist es wenn man das vorhandene Verzeichnis mit dem Namen *linux* (die alten Kernel-Sourcen) erst wegkopiert.

```
tar xzf linux-2.4.3.tar.gz
```

ist der Befehl um das Archiv zu entpacken.
Falls man schon eine recht aktuelle Version (z.B. 2.4.2) der Linux Kernel Sourcen auf dem System hat, lohnt es sich nicht, das komplette Archiv für die Version 2.4.3 runterzuladen, die doch recht groß ist (26MB). Am besten ist es dann, sich den Patch runterzuladen. Das Einspielen eines Patches wird mit dem folgenden Befehl gemacht:

```
gzip -dc patch-2.4.3.gz | patch -p0
```

Um sicher zu gehen, dass man auch die gewünschte Version des Kernels entpackt hat, schaut man sich im Verzeichnis *linux* die Datei *Makefile* an.

```
> head -4 /usr/src/linux/Makefile
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 2
EXTRAVERSION = -ac13
```

Diese Ausgabe entspricht der Version 2.4.2.

2.3 Konfiguration

Über die Konfiguration wird der Kernel an das jeweilige Zielsystem angepasst. Hier kann der Benutzer entscheiden, welche Features/Treiber in den Kernel eingebaut werden sollen. Die meisten Komponenten lassen sich hierbei auch als Module übersetzen (Option *m*). Zu jeder Option gibt es einen kurzen Hilfetext. Mehr Dokumentation steht im *Documentation* Unterverzeichnis zur Verfügung.

Die Einstellungen werden im Kernel-Verzeichnis in der Datei *.config* festgehalten. Da das (komplette) Konfigurieren einige Zeit in Anspruch nimmt, kann man eben dieses File auch einfach von einer Vorgänger-Version kopieren oder über eines der Konfigurationstools importieren. Diese Tools werden im Folgenden erklärt.

2.3.1 make config / make oldconfig

Häufig wird *make config* auch als die Text-Adventure Variante der Kernel-Konfiguration bezeichnet. Hier stellt ein shell-script zu jeder Option eine Frage, und der Benutzer kann über String-Eingaben die entsprechende Antwort wählen. Bei einer leeren Eingabe wird eine default-Antwort verwendet.

Auszug:

```
/usr/src/cvs/linux > make config
rm -f include/asm
( cd include ; ln -sf asm-sparc64 asm)
/bin/sh scripts/Configure arch/sparc64/config.in
#
# Using defaults found in .config
#
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers
   (CONFIG_EXPERIMENTAL) [Y/n/?]
*
* Loadable module support
*
Enable loadable module support (CONFIG_MODULES) [Y/n/?]
  Set version information on all symbols for modules
   (CONFIG_MODVERSIONS) [Y/n/?]
  Kernel module loader (CONFIG_KMOD) [Y/n/?]
*
* General setup
*
UltraSPARC-III bootbus i2c controller driver
   (CONFIG_BBC_I2C) [N/y/m/?]
Symmetric multi-processing support (CONFIG_SMP) [N/y/?]
```

Mit *make oldconfig* wird die vorhandene Kernel Konfiguration (*.config*) verglichen mit den Möglichkeiten des neuen Kernels, und man muss, im Stil von *make config*, nur auf die Sachen antworten, die durch diesen Kernel neu sind. Diese Methode ist die einfachste wenn man schon eine funktionierende Konfiguration hat.

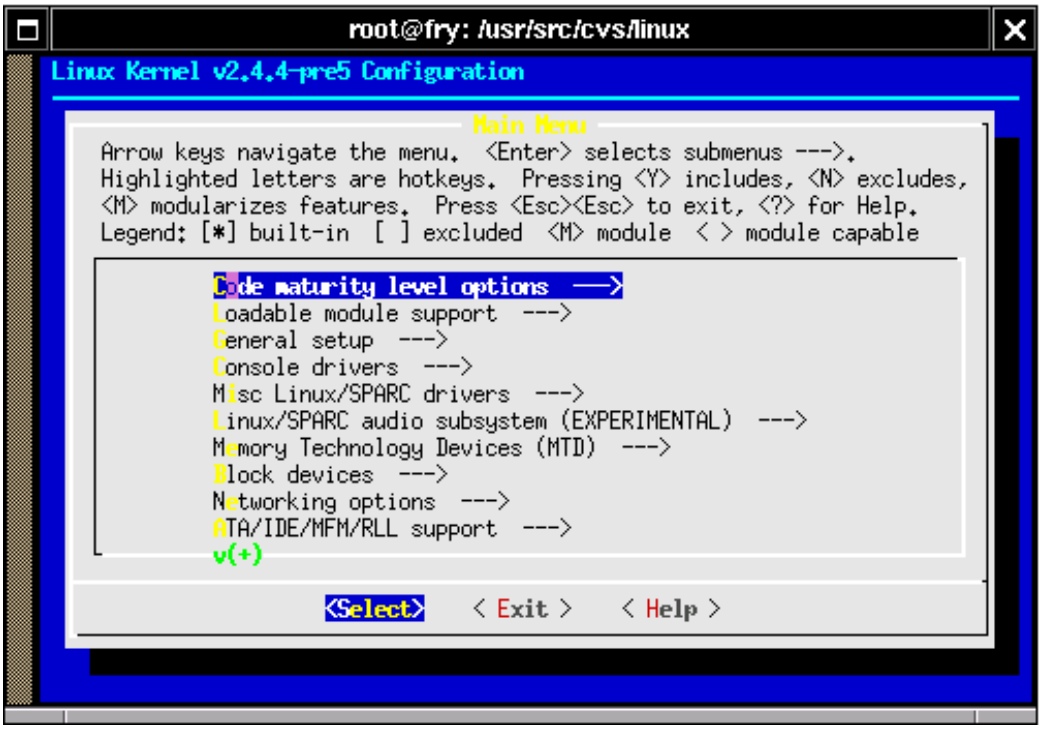

Abbildung 2.1: make menuconfig

2.3.2 make menuconfig

Etwas einfacher als *make config* erweist sich *make menuconfig*. Diese Konfigurationsvariante basiert auf der *ncurses* Bibliothek und lässt sich recht einfach bedienen. Man kann hier ganz einfach mit Tab und den Cursortasten durch das Menü navigieren (ähnlich den typischen *BIOS* Setups).

2.3.3 make xconfig

Am komfortabelsten ist sicher *make xconfig*. Wer seinen Kernel mit der Maus konfigurieren möchte, braucht allerdings eine funktionierende *X11* und *tcl/tk* Installation.

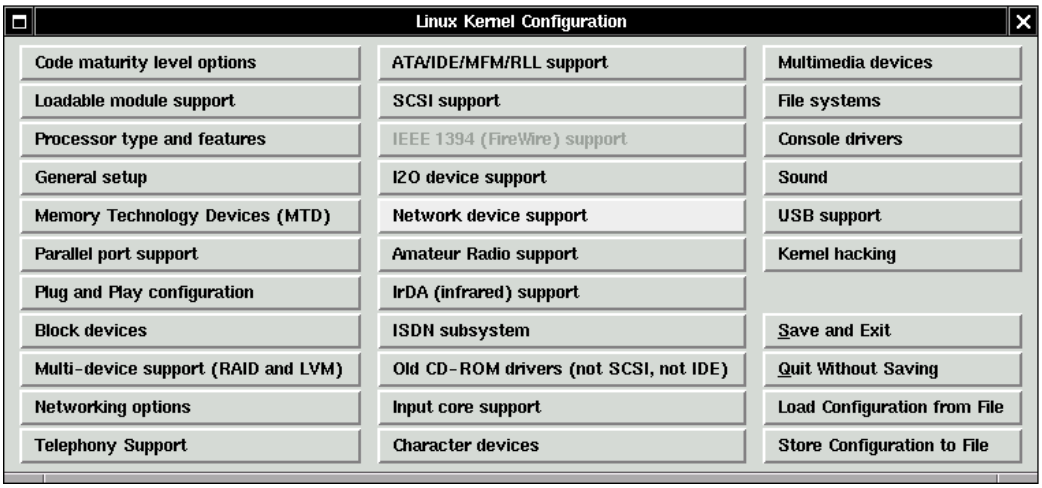

Abbildung 2.2: make xconfig

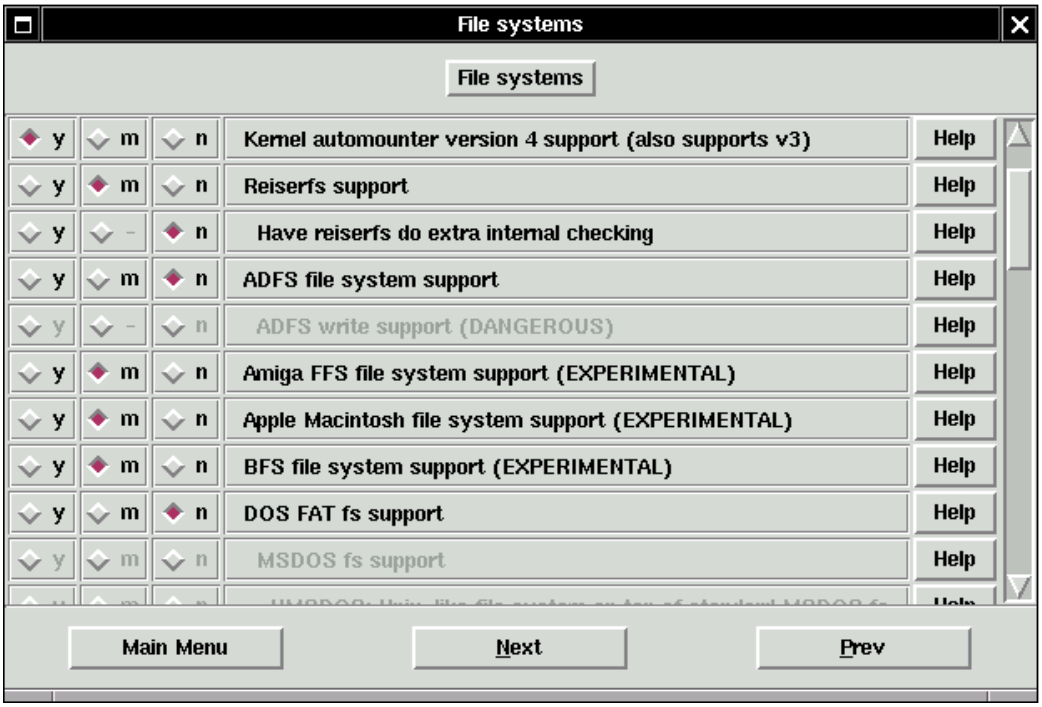

Abbildung 2.3: make xconfig - File systems

2.4 Kompilieren

Das Kompilieren des Kernels erweist sich als erstaunlich einfach. Üblicherweise reicht hier ein

```
make dep && make clean && make bzImage && make modules
```

2.4.1 make dep

Mit *make dep* werden die Abhängigkeiten (*dependencies*) richtig gesetzt.

2.4.2 make clean

Mit *make clean* wird im *linux*-Verzeichnis aufgeräumt. Das heißt fast alle Spuren einer früheren Kompilation werden entfernt. Wirklich alles wird gelöscht mit *make mrproper*. Aber Vorsicht: auch die Datei *.config* ist dann weg und man muss alles noch mal konfigurieren...

2.4.3 make bzImage

Hier findet die eigentliche Arbeit für den Prozessor statt. Hier wird alles übersetzt und gelinkt, was nicht Modul sein soll. Effektiv findet das Gleiche statt wie wenn man nur *make* eingibt, aber zusätzlich wird der neu entstehende Kernel auch noch gepackt und hoffentlich kleiner sein als die magische Grenze 640KB.

Den neuen Kernel findet man nun direkt im *linux*-Verzeichnis als Datei mit den Namen *vmlinux*. Diese Datei ist aber viel zu groß und deshalb ist unter *arch/i386/boot* der Kernel komprimiert in der Datei *bzImage*.

2.4.4 make modules

Mit *make modules* werden die Module generiert.

2.5 Installation

Sind die Module und das Kernel-Image fertiggestellt, müssen sie noch entsprechend ins System integriert werden, damit mit sie beim nächsten Systemstart auch verwendet werden. Vor der Installation eines neuen Kernels empfiehlt es sich, dafür zu sorgen, dass man ein altes funktionierendes Kernel-Image bootfähig hält, für den Fall dass das neue doch nicht booten möchte.

2.5.1 Installation der Kernels mit *lilo*

Das Kernel-Image muss nun an die in */etc/lilo.conf* angegebene Stelle im Filesystem kopiert werden. Typischerweise ist das */boot/bzImage*. Nun muss noch die *loading map* aktualisiert werden - das geschieht mit einem simplen *lilo* Aufruf. Außerdem sollte man noch die Datei *System.map* nach */boot/System.map* kopieren.

2.5.2 Installation der Module

Um die Module zu installieren reicht ein einfaches *make modules_install*

2.6 Kurzfassung

Hier nochmal alle nötigen Befehle kurz zusammengefasst:

```
> cd /usr/src
> wget ftp://ftp.de.kernel.org/pub/linux/kernel/\
>v2.4/linux-2.4.3.tar.gz # download des kernels
> tar xzf linux-2.4.3.tar.gz # entpacken
> cd linux
> make menuconfig # konfigurieren
> make dep
> make clean
> make bzImage # kernel bauen
```

```
> make modules # module
> make modules_install # module installieren
> cp arch/i386/boot/bzImage /boot # kernel an die \
>richtige(TM) stelle kopieren
> cp System.map /boot
> lilo -v # neuen kernel in den bootsector schreiben
> reboot # daumen druecken
```

Kapitel 3

Neuerungen in der Version 2.4

Diese neueste Version des Linux Kernels ist in den meisten Distributionen noch nicht enthalten, außer man besitzt eine sehr aktuelle Version.

Deshalb hier ein paar Gründe, doch auf eine 2.4.x Version umzusteigen:

Performance Durch eine erhebliche Überarbeitung der VM und des Speichermanagements ist diese neueste Linux Version noch performanter als die früheren Versionen.

USB Die USB Unterstützung ist von der Entwicklerversion in die stabile Version übernommen worden, und USB Geräte wie Keyboard, Maus, Kameras und PDAs werden unterstützt.

ISA Plug'n'Play wurde in den Kernel integriert.

IP firewalling und NAT wurde komplett überarbeitet. Mit der neuen Version sind auch statefull firewall Regeln möglich.

ECN Explicit Congestion Notification - Erweiterung von TCP/IP, so dass die Pakete einen Ersatzweg finden bei Verstopfung an bestimmten Punkten im Netz. Als dies soll passieren ohne dass der Router, an dem die Verstopfung auftritt, die Pakete verwerfen muss. Die Ersatzroute soll schon davor klar sein. (RFC 2481)

Außerdem gibt es noch viele Änderungen, die eher für den Einsatz von Linux im *Enterprise/Server* Bereich gedacht sind.

Journaling Filesystem Mit reiserfs ist eines von mehreren für Linux verfügbaren Journaling Filesystemen jetzt Bestandteil des Kernels.

NFSv3 Version 3 dieses weit verbreiteten Netzwerkfilesystems wird unterstützt.

LVM Mit Logical Volume Management ist es möglich, logische Partitionen auf mehrere physikalische Partitionen zu verteilen.