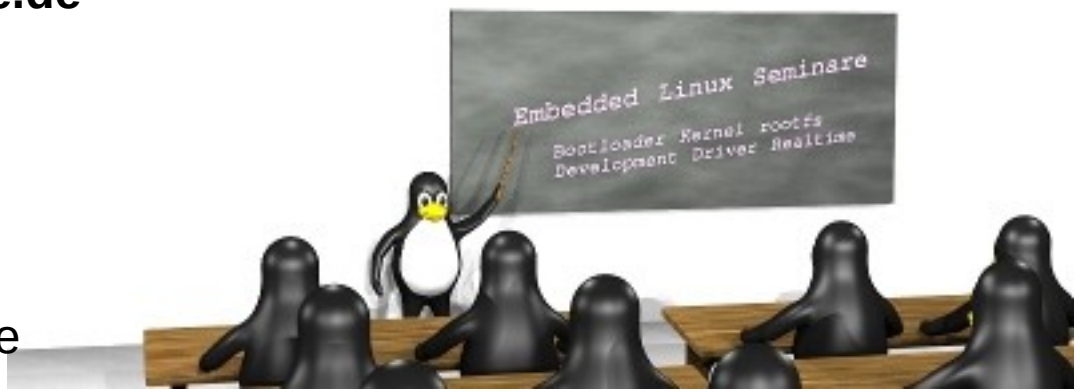
Embedded-Linux-Seminare

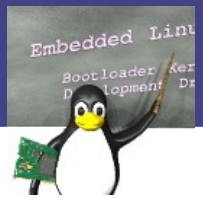
Linux Kernel

http://www.embedded-linux-seminare.de

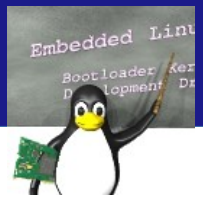
Diplom-Physiker Peter Börner
Spandauer Weg 4
37085 Göttingen

Tel.: 0551-7703465
Mail: info@embedded-linux-seminare.de





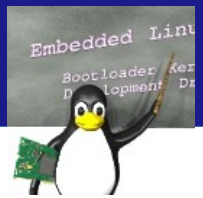
Kopier-Rechte



Inhalt

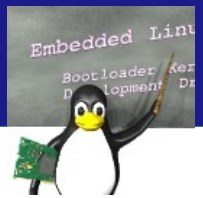
Kompilieren und Bootvorgang

- Linux kernel sources
- Kernel configuration
- Compiling the kernel
- Overall system startup
- Linux device files
- Cross-compiling the kernel



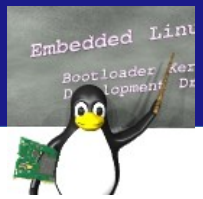
Kernel Quellen

- Die offiziellen Kernel Quellen sind zu finden unter http://www.kernel.org
- Es gibt eigene von Entwickler Gemeinden gepflegte Kernel Versionen für bestimmte Zwecke oder Architekturen
  - Architekturen (ARM, MIPS, PowerPC, etc.),
  - Treiber (I2C, SPI, USB, PCI, network, etc.),
  - Echtzeit, etc.
  - Es werden keine offiziellen Releases herausgegeben, nur Entwicklungs Versionen

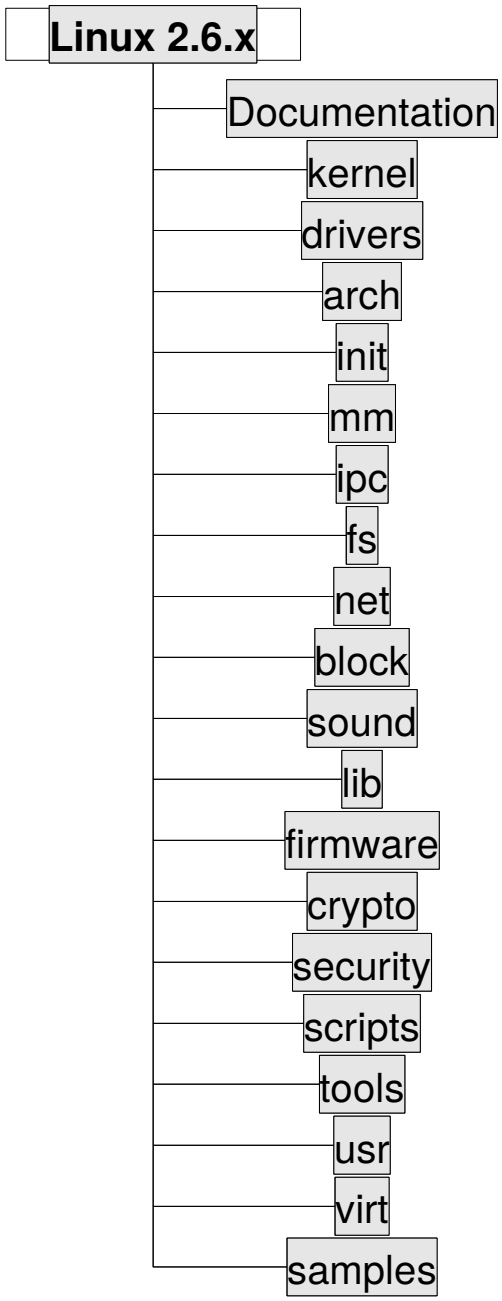


Linux Kernel Größe (1)

- Linux 5.0 Quellcode Größen:
  - Ausgepackt: 493 MB
    67,353 Dateien, ca. 26 Millionen Zeilen Code
  - xv komprimierter Tarball: 101 MB
- Der kompilierte komprimierte Kernel liegt für Embedded Systeme bei ca. 7 MB..
- Warum ist der Quellcode zu groß?
  - Der Quellcode enthält unzählige Treiber für Geräte, Filesysteme, Netzwerk und Code für verschiedenste Architekturen.
  - Der Kern des Kernels (Scheduler, Speicher Verwaltung...) ist sehr klein.



Kernel Quellcode Aufbau



Dokumentation, sehr lesenswert

Der eigentliche Kernel (Scheduler, etc)

Treiber

Architektur spezifischer Code

Initialisierung

Speicher Verwaltung

Interprozess Kommunikation

Filesystem Treiber

Netzwerk Treiber

Block Devices

Sound Treiber

Gemeinsam genutzte Funktionen

Firmware

Verschlüsselung

Sicherheit

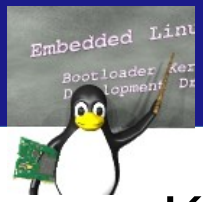
Skripte zum Bauen des Kernels

Tools für den Bauvorgang

Userland Code (Gpios)

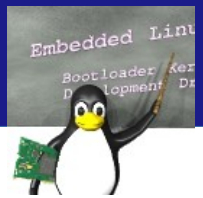
Virtualisierung (kvm)

Beispiele



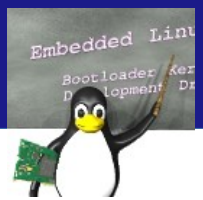
Linux Quellen holen

- Komplette Tarballs
  - Enthalten den kompletten Quellcode
  - Download und Auspacken dauert ein wenig länger
- Patches zwischen einzelnen Versionen
  - Setzt eine Basis Version voraus (z.B. 4.20)
  - Patches müssen in der richtigen Reihenfolge eingespielt werden
  - Schneller Download
  - Beispiel
    https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/patch-4.20.2.xz
    (4.20 auf 4.20.2)
  - Patches werden ebenfalls für Erweiterungen genutzt
    - Andere Architektur
    - Echtzeiterweiterung



Der Patch Befehl

- Der `patch` Befehl ändert Dateien im aktuellen Verzeichnis:
  - Ändert Dateien
  - Erzeugt oder löscht Dateien und Verzeichnisse
- `patch` Benutzung:
  - `patch -p<n> < diff_file`
  - `cat diff_file | patch -p<n>`
  - `bxzcat diff_file.xz | patch -p<n>`
  - `n`: Anzahl der zu ignorierenden Verzeichnislevel in der Pfad Angabe
- Ein Patch kann mit der Option -R rückgängig gemacht werden
- Mit der Option –dry-run kann ein Testlauf durchgeführt werden



Aufbau einer Patch Datei

Eine Patch Datei wird durch einen `diff` Befehl erzeugt

```
diff --git a/Makefile b/Makefile
index 7a2a9a175756..4ba3dd0bf35d 100644
--- a/Makefile
+++ b/Makefile
@@ -1,7 +1,7 @@
# SPDX-License-Identifier: GPL-2.0
 VERSION = 4
 PATCHLEVEL = 20
-SUBLEVEL = 0
+SUBLEVEL = 2
 EXTRAVERSION =
 NAME=Shy Crocodile
```

`diff` Befehlszeile

Datei Info

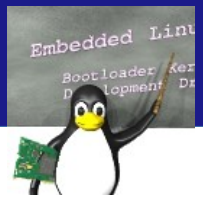
Zeilennummer in Datei

Kontext Info: 3 Zeilen vor der Änderung
Hilfreich, wenn die Zeilennummer nicht mehr ganz passen

Zeile(n) wird entfernt

Zeile(n) wird hinzugefügt

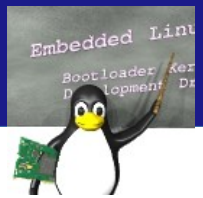
Kontext Info: Zeilen hinter der Änderung



Linux Patch einspielen

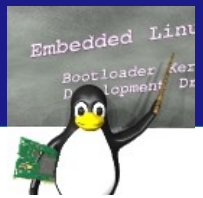
- Linux Patches...
  - Gehören immer zu der `x.y.<z-1>` Version
    Als xz Dateien vorhanden
  - Sind für `n=1` erzeugt worden
- Linux Patch Beispiel:

```
cd linux-4.20
xzcat ../patch-4.20.2.xz | patch -p1
cd ..;
mv linux-4.20 linux-4.20.2
```



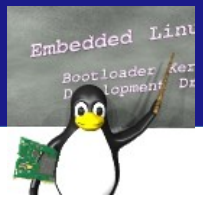
Kernel Konfiguration (1)

- Der Kernel enthält unzählige Treiber, Filesysteme, Netzwerkprotokolle und Architekturen
- Diese müssen konfiguriert werden für
  - die vorhandene Hardware
  - die benötigten Eigenschaften des Kernels
- Viele Optionen beeinflussen den Bauvorgang des Kernels
- Die Konfiguration ist in einer einzigen Textdatei gespeichert
- Die Konfigurationsdatei `.config` befindet sich im Wurzelverzeichnis des Quellcode Pfads
- Der Aufbau ist einfach
  - Attribut = Wert
  - # Attribut is not set



Kernel Konfiguration (2)

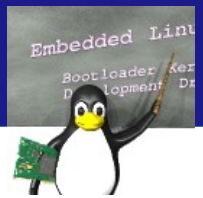
- Die Konfiguration kann mit einem Editor geändert werden
  !!! Nur für Experten, da es viele Abhängigkeiten gibt
- Andere Varianten sind
  - `make config` Frage und Antwortspiel
  - `make menuconfig` ASCII Grafik, braucht ncurses
  - `make gconfig` GTK Grafik, braucht GTK
  - `make xconfig` Qt3 Grafik, braucht Qt3 Lib
  - Targets des zentralen Makefiles.
    `make help` zeigt alle nutzbaren Targets an



make xconfig

`make xconfig`

- Häufig verwendetes GUI zur Kernelkonfiguration
- Hilfe gibt einige einführende Hinweise unter
  `help -> introduction`
- Suchmöglichkeit
- benötigt Qt Entwicklungsumgebung
  z.B. `libqt-mt-dev`, `g++`



make xconfig

qconf

File Option Help

Option	Name
Code maturity level options	
General setup	
☑ Configure standard kernel features (for small systems)	EMBEDDED
Loadable module support	
System Type	
Intel PXA2xx Implementations	
☑ Toshiba e7xx / e8xx	ARCH_ESERIES
☑ Asus 620/620BT	MACH_A620
☑ hp iPAQ h1910	ARCH_H1900
☑ hp iPAQ h2200	ARCH_H2200
☑ hp iPAQ h3900	ARCH_H3900
☑ hp iPAQ h4000	MACH_H4000
☑ hp iPAQ h5400	ARCH_H5400
☐ Dell Axim X5	ARCH_AXIMX5
☐ Dell Axim X3 (non-functional)	ARCH_AXIMX3
☑ RoverP1 (Mitac Mio 336)	ARCH_ROVERP1
☑ RoverP5+	ARCH_ROVERP5P
Linux As Bootloader	
Compaq/iPAQ Options	
General setup	
PCMCIA/CardBus support	
Generic Driver Options	
Parallel port support	
Memory Technology Devices (MTD)	
RAM/ROM/Flash chip drivers	
Mapping drivers for chip access	
Self-contained MTD device drivers	
NAND Flash Device Drivers	
Plug and Play support	

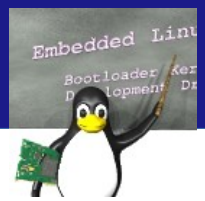
Option	Name
..	
⊙ iPAQ H2200 PCMCIA	H2200_PCMCIA
⊙ iPAQ H2200 MediaQ 1178 LCD	H2200_LCD
☐ iPAQ H2200 battery interface	H2200_BATTERY
⊙ iPAQ H2200 touchscreen driver	H2200_TS
⊙ iPAQ H2200 hardware audio control	H2200_AUDIO

hp iPAQ h2200 (ARCH_H2200)

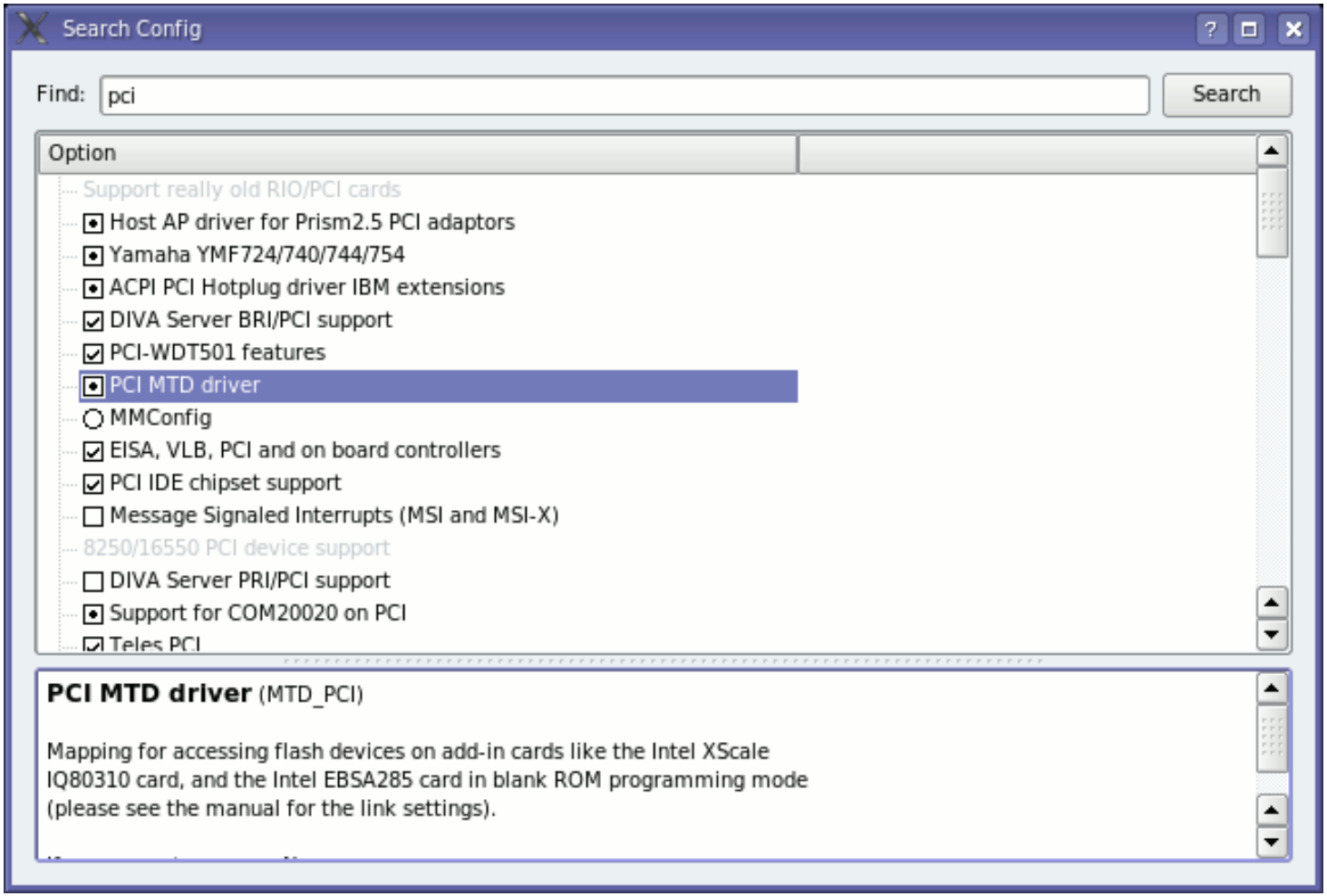
type: boolean
prompt: hp iPAQ h2200
  dep: ARCH_PXA
select: PXA25x
  dep: ARCH_PXA

defined at arch/arm/mach-pxa/h2200/Kconfig:1

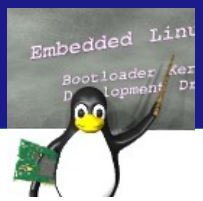
This enables support for HP iPAQ H22xx series of handhelds.
There are a number of H22xx-specific drivers under this submenu:
pcmcia, lcd, battery, touchscreen



make xconfig Suche



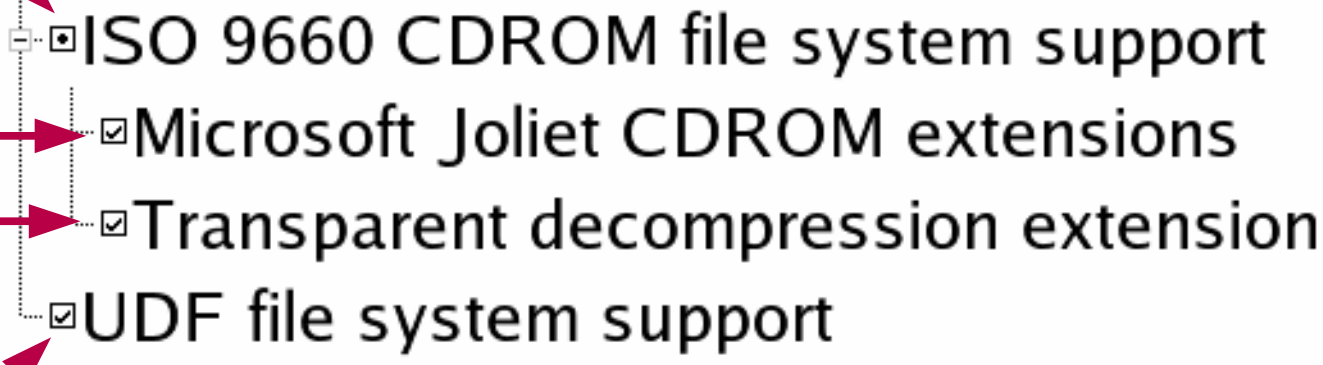
Sucht nach Parameter und erlaubt die gefundenen Parameter direkt zu ändern



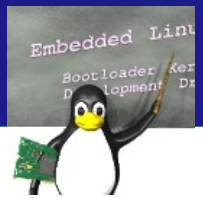
Kernel Konfigurations Optionen

Als nachladbares Modul erstellen
`CONFIG_ISO9660_FS=m`

Treiber Optionen
`CONFIG_JOLIET=y`
`CONFIG_ZISOFS=y`



Statisch in den Kernel gelinkt
`CONFIG_UDF_FS=y`

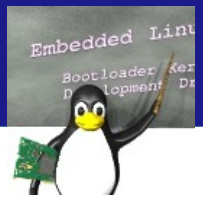


Zugehöriger Ausscnitt der .config Datei

```
#
# CD-ROM/DVD Filesystems
#
CONFIG_ISO9660_FS=m
CONFIG_JOLIET=y
CONFIG_ZISOFS=y
CONFIG_UDF_FS=y
CONFIG_UDF_NLS=y

#
# DOS/FAT/NT Filesystems
#
# CONFIG_MSDOS_FS is not set
# CONFIG_VFAT_FS is not set
CONFIG_NTFS_FS=m
# CONFIG_NTFS_DEBUG is not set
CONFIG_NTFS_RW=y
```

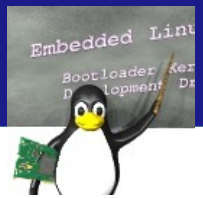
Name des Konfigurationsbereichs
(hilfreich um die Settings zu finden)

Alle Parameter habe als Prefix
`CONFIG_`



Kernel Options Abhängigkeiten

- Es gibt Abhängigkeiten zwischen Kernel Optionen
- Z.B.: Ein Treiber für eine Netzwerkkarte setzt vorraus, dass der Netzwerkstack aktiviert ist
- Zwei Arten von Abhängigkeiten
  - *depends on* Abhängigkeiten.In diesem Fall ist Option A, welche auf Option B beruht, nicht sichtbar bis Option B aktiviert ist.
  - *select* Abhängigkeiten. Wenn Option A auf Option B beruht, so wird Option B automatisch aktiviert, wenn Option A aktiviert wird.
  - *make xconfig* zeigt alle Optionen, auch die nicht konfigurierbaren. Diese sind dann ausgegraut.



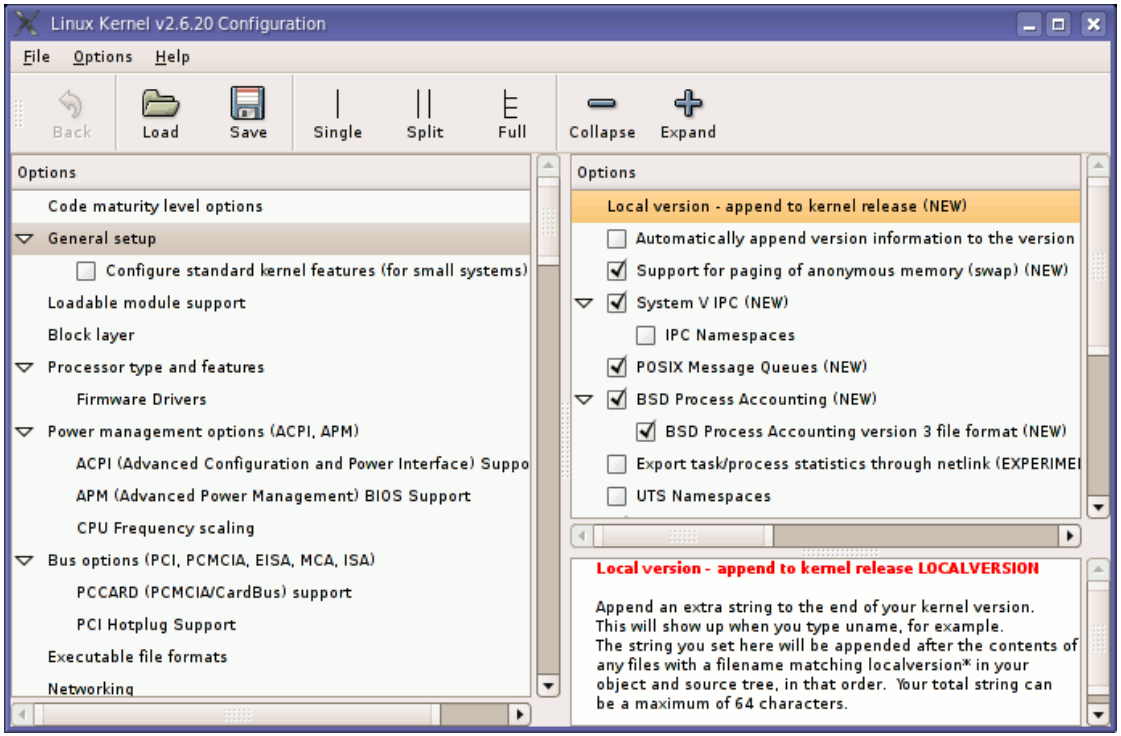
make gconfig

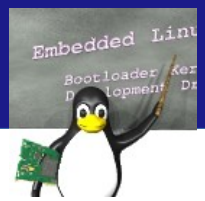
`make gconfig`

GTK basiertes Interface.

Functionalität ähnlich zu `make xconfig`.

Benötigt libglade





make menuconfig

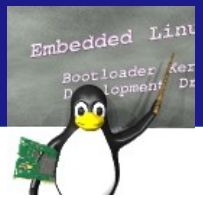
```
Linux Kernel v2.6.19 Configuration
                    Processor type and features
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

      [ ] Symmetric multi-processing support
          Subarchitecture Type (PC-compatible)  --->
          Processor family (Pentium-Pro)  --->
      [*] Generic x86 support
      [ ] HPET Timer Support
          Preemption Model (No Forced Preemption (Server))  --->
      [ ] Local APIC support on uniprocessors
      [ ] Machine Check Exception
      < > Toshiba Laptop support
      < > Dell laptop support
      [ ] Enable X86 board specific fixups for reboot
      <M> /dev/cpu/microcode - Intel IA32 CPU microcode support
      < > /dev/cpu/*/msr - Model-specific register support
      <*> /dev/cpu/*/cpuid - CPU information support
          Firmware Drivers  --->
      v(+)

                  <Select>    < Exit >    < Help >
```

`make menuconfig`

Sinnvoll wenn Grafik nicht vorhanden ist.

enötigt

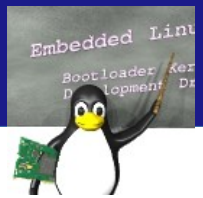
`libncurses-dev`



make oldconfig

make oldconfig

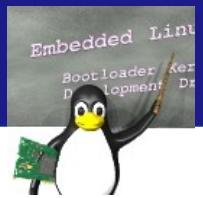
- Wird häufig gebraucht
- Nützlich beim Upgrade auf einen neuen Kernel
- Liest vorhanden .config Datei ein
- Warnt bei nicht mehr vorhandenen Konfigurations Parametern
- Fragt bei neu hinzugekommenen Konfigurations Parametern die gewünschten Settings ab
- Bei manueller Änderung der .config Datei nützlich, um Abhängigkeiten aufzulösen



make allnoconfig

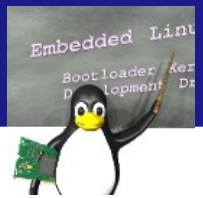
`make allnoconfig`

- Gleichbedeutend wie `make config` und beantworten aller Yes/No Fragen mit n.
- Alle streng empfohlenen Settings werden mit y beantwortet
- Nützlich um eine minimale Konfiguration für eine Embedded System zu erzeugen
- Das Gegenteil existiert auch: `make allyesconfig`



Weitere automatische Konfigurationen

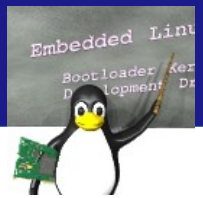
- `make allmodconfig`
  - Wenn möglich wird die Option für den Bau eines nachladbaren Moduls gesetzt
- `make alldefconfig`
  - Setzt alle Optionen auf Defaultwerte
- `make defconfig`
  - Setzt die Werte entsprechend einer Konfigurationsdatei, die für die gewählte Architektur in dem entsprechenden Unterverzeichnis liegt.
  - Sinnvoll für Embedded-Systeme.
  - Es kann für eine bestehende Hardware eine Default-Konfiguration angelegt und weiterverwendet werden



Default Config Dateien

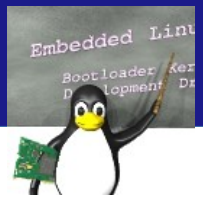
```
assabet_defconfig       integrator_defconfig    mainstone_defconfig
badge4_defconfig        iq31244_defconfig       mx1ads_defconfig
bast_defconfig          iq80321_defconfig       neponset_defconfig
cerfcube_defconfig      iq80331_defconfig       netwinder_defconfig
clps7500_defconfig      iq80332_defconfig       omap_h2_1610_defconfig
ebsa110_defconfig       ixdp2400_defconfig      omnimeter_defconfig
edb7211_defconfig       ixdp2401_defconfig      pleb_defconfig
enp2611_defconfig       ixdp2800_defconfig      pxa255-idp_defconfig
ep80219_defconfig       ixdp2801_defconfig      rpc_defconfig
epxa10db_defconfig      ixp4xx_defconfig        s3c2410_defconfig
footbridge_defconfig    jornada720_defconfig    shannon_defconfig
fortunet_defconfig      lart_defconfig          shark_defconfig
h3600_defconfig         lpd7a400_defconfig      simpad_defconfig
h7201_defconfig         lpd7a404_defconfig      smdk2410_defconfig
h7202_defconfig         lubbock_defconfig       versatile_defconfig
hackkit_defconfig       lusl7200_defconfig
```

`arch/arm/configs` Beispiele



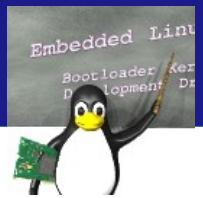
Konfigurationen rückgängig machen

- Ein häufiges Problem:
  - Zahlreiche Parameter wurden geändert und der Kernel arbeitet nicht mehr
  - Aber was tun, wenn man sich nicht an alle Änderungen erinnert?
    `> cp .config.old .config`
  - Alle Konfiguration tools des Kernels (`xconfig`, `menuconfig`, `allnoconfig`...) sichern die alten Einstellungen in `.config.old`



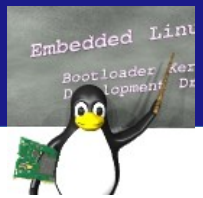
Konfiguration verschiedener Architekturen

- Die Konfigurations Optionen sind abhängig von der gewählten Architektur
  - Einige Optionen sind sehr Architektur bezogen
  - Die meisten Optionen (Netzwerk, Filesystem, etc) sind in allen Architekturen vorhanden
- Ohne weiter Angaben geht das Makefile davon aus, dass für die Architektur des Host Rechners gebaut werden soll
- `make ARCH=architektur xconfig`
  verändert die Konfigurations Optionen entsprechend
  - Es lohnt sich mal auszuprobieren
    `make ARCH=arm xconfig`
    `make ARCH=powerpc xconfig`



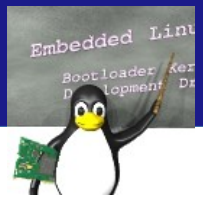
Übersicht über die Kernel Optionen (1)

- General setup
  - *Prompt for development/incomplete code* erlaubt auch Features oder Treiber zu aktivieren, die noch nicht als stabil gekennzeichnet sind.
  - *Local version - append to kernel release* ermöglicht es dem Kernel einen eigenen Namensanhang zu geben. Sinnvoll zur Kennzeichnung für welche Hardware er gedacht ist oder ob er z.B. Echtzeit fähig ist.
  - *Support for swap*, wird in Embedded Systemen nicht benötigt
  - *Configure standard kernel features (for small systems)* erlaubt es die Größe des Kernels durch Einschränkung der Features zu reduzieren.



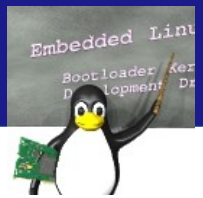
Übersicht über Kernel Optionen (3)

- Kernel features
  - Tickless system erlaubt dynamische Zeitintervalle anstatt feste zyklische Timer Events
  - High resolution timer support. Erlaubt mit Timern mit höherer Auflösung als die Timer-Ticks zu arbeiten
  - Preemptible kernel aktiviert mehr oder minder stark die Unterbrechbarkeit von Prozessen im Kernel
- Power management
  - Globale Power Management Option
  - Suspend to RAM, CPU frequency scaling, CPU idle control, suspend to disk



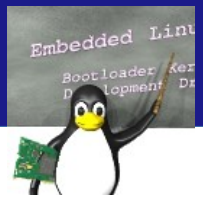
Übersicht über Kernel Optionen (4)

- Networking support
  - Netzwerk Stack
  - Netzwerk Optionen
    - Unix sockets, für Inter-Prozess Kommunikation
    - TCP/IP Protokoll mit Optionen für Multicast, Routing, Tunneling, Ipsec, Ipv6, etc.
    - Andere Protokolle such as DCCP, SCTP, TIPC, ATM
    - EthernetBbridging, QoS, etc.
  - Unterstützung für andere Arten von Netzwerk
    - CAN bus, Infrared, Bluetooth, Wireless stack, WiMax stack, etc.



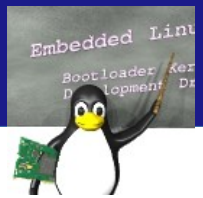
Übersicht über Kernel Optionen (5)

- Device drivers
  - MTD ist das Subsystem für Flash (NOR, NAND, OneNand, battery-backed memory, etc.)
  - Parallel Port Unterstützung
  - Block Devices, einige verschiedene Block Treiber, wie loopback, NBD, etc.
  - ATA/ATAPI, Unterstützung für IDE disk, CD-ROM
  - SCSI
    - Der SCSI Kern, wird auch für USB Speichermedien, SATA und PATA Speichermedien gebraucht.



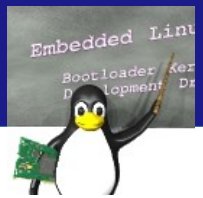
Übersicht über Kernel Optionen (6)

- Device drivers (Fortsetzung)
  - SATA and PATA, für aktuelle Festplatten
  - Netzwerk Controller Unterstützung. Ethernet, Wireless, aber auch PPP
  - Input Device Unterstützung für Tastaturen, Mäuse, Joysticks, Touchscreen, etc
  - Character Devices
    - Serielle Schnittstellen Treiber
    - PTY Treiber, werden von SSH oder Telnet gebraucht
  - I2C, SPI, 1-wire
  - Hardware Monitoring Unterstützung, Infrastrukture und Treiber für Thermal Sensoren



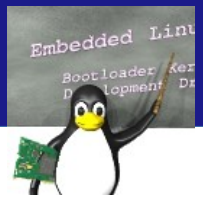
Übersicht über Kernel Optionen (7)

- Device drivers (Fortsetzung)
  - Watchdog
  - Multimedia Unterstützung, enthält Video für Linux V4L und DVB Subsysteme, für Video Capture, Webcams, AM/FM Karten, DVB Adapter
  - Graphics Unterstützung, Infrastruktur und Treiber für Framebuffer
  - Sound Card Unterstützung, OSS und ALSA Sound Infrastruktur



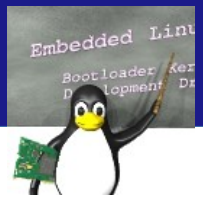
Übersicht über Kernel Optionen (8)

- Device drivers (Fortsetzung)
  - USB Unterstützung
    - Infrastruktur
    - Host Controller Treiber
    - Device Treiber
    - Gadget Controller Treiber
    - Gadget Treiber, um das Embedded System als Mass-Storage Device, Serielle Schnittstelle oder Ethernet Adapter erscheinen zu lassen
  - MMC/SD/SDIO
  - LED
  - Real Time Clock
  - Spannungs- und Strom-Regler
  - Staging Treiber



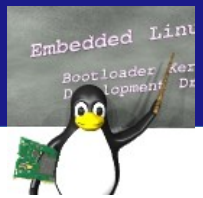
Übersicht über Kernel Optionen (9)

- File Systeme
  - ext2, ext3, ext4
  - XFS, JFS, GFS2, OCFS2, Btrfs
  - CD-ROM: ISO9660, UDF
  - DOS/Windows: FAT and NTFS
  - Pseudo Filesysteme: proc and sysfs
  - Flash Filesysteme wie JFFS2, UBIFS, SquashFS, cramfs
  - Netzwerk Filesystems: z.B. NFS und SMB/CIFS
- Kernel Hacking
  - Debugging Möglichkeiten für Kernel Entwickler



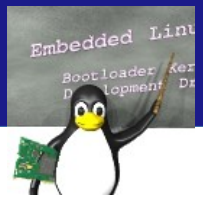
Kernel Compilieren

- `make`
  - im Hauptverzeichnis des Kernel Quellcode Pfads
  - Wenn mehrere CPUs vorhanden sind kann `make -j 4` genutzt werden
- Erzeugt
  - `vmlinux`, das unkomprimierte Kernel Image. Dies kann zum Debuggen genutzt werden. Es kann nicht gebootet werden.
  - `arch/<arch>/boot/*Image`, der endgültige bootfähige Kernel
    - `bzImage` für x86
    - `zImage` für ARM
    - `vmImage.gz` für Blackfin
    - uImage für U-Boot
    - etc.
  - Die nachladbaren Kernel Module sind über den Quellcode Pfad verteilte Dateien mit der Endung `.ko`



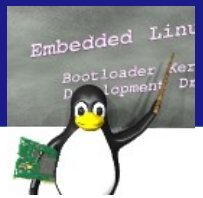
Kernel Installation

- `make install`
  - Installiert den Kernel in das Host System.
- Installiert
  - `/boot/vmlinuz-<version>`
    der komprimierten Kernel aus `arch/<arch>/boot`
  - `/boot/System.map-<version>`
    Datei mit den Kernel Symbol Adressen
  - `/boot/config-<version>`
    Kernel Konfiguration der installierten Version
- Konfiguriert meist den Bootloader, um den neuen Kernel starten zu können



Module Installation

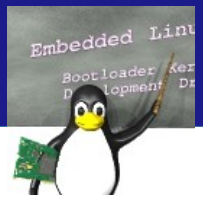
- `make modules_install`
  - Installiert die nachladbaren Module nach /lib/modules/<version>/kernel
  - Die Struktur des Quellcodepfads wird beibehalten
  - `modules.alias`
    Aliase für die nachladbaren Module, z.B.:
    `alias sound-service-?-0 snd_mixer_oss`
  - `modules.dep`
    Modul Abhängigkeiten
  - `modules.symbols`
    Zugehörigkeit der Module zu den Symbolen



Kernel Aufräumen

- `make clean`
  Löscht die beim Bauen erzeugten Dateien
- `make mrproper`
  Versetzt den Kernel Quellcodepfad fast wieder in den Ausgangszstand. Es werden alle generierten Dateien gelöscht.
  !!! Achtung: Auch .config wird gelöscht

- `make distclean`
  Löscht auch die Backup Dateien, die der Editor hinterlassen hat und die Patch Reject Dateien

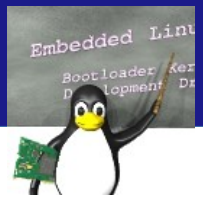


Cross-Compilieren des Kernels (1)

Erstellt den Linux Kernel für eine andere Architektur als die des Host Rechners

- Schneller als den Bauvorgang auf einem langsamen Embedded System laufen zu lassen
- Besser nutzbar für Kernel- und Treiberentwicklungen
- Die Cross-Compiler erhalten als Kennzeichnung eine Präfix, der das Zielsystem beschreibt, z.B.:

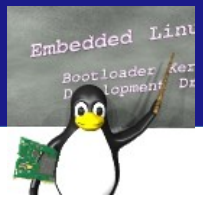
```
mips-linux-gcc
m68k-linux-uclibc-gcc
arm-linux-gnueabi-gcc
```



Cross-Compilieren des Kernels (2)

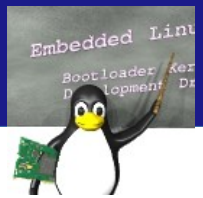
Die CPU und der Cross-Compiler werden durch die Variablen `ARCH` und `CROSS_COMPILE` im zentralen `Makefile` `festgelegt`.

- Das `Makefile` definiert `CC = $(CROSS_COMPILE)gcc`
- Der einfachste und gebräuchliste Weg diese Variablen zu setzen ist, sie als Argumente an Make zu übergeben:
  - `make ARCH=cpu CROSS_COMPILE=präfix`
    z.B.
    `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-gcc`
- Ein anderer Weg ist, das Makefile zu verändern.
- Der Cross-Compiler kann auch beim Konfigurien bei den Generellen Einstellungen gesetzt werden. Dies setzt CONFIG_CROSS_COMPILE
- Der Cross-Compiler muss im Suchpfad zu finden sein



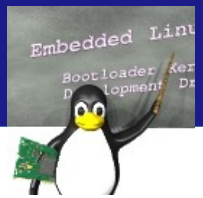
Cross-Compilieren des Kernels (3)

- `make ARCH=cpu CROSS_COMPILE=präfix`
- Die Datei
  `arch/<arch>/boot/zImage`
  an den richtigen Platz kopieren
- `make install INSTALL_PATH=pfad`
  tut das Gleiche, z.B.:
  `make install INSTALL_PATH=/tftpboot`
- `make INSTALL_MOD_PATH=<dir> modules_install`
  kopiert die Module nach
  `<dir>/lib/modules/`



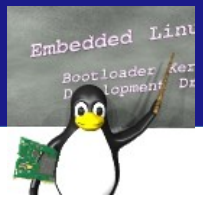
Nachladbare Kernel Module

- Module: Fügen Funktionalitäten zum Kernel zur Laufzeit hinzu (Filesystem, Zugriff auf Hardware,...)
- Können jederzeit geladen oder entladen werden
- Module mache das Entwickeln eigener Treiber einfacher, da sie zum Testen geladen, Debuggt und wieder entladen werden können.
- Nützlich um den eigentlichen Kernel klein zu halten, aber eine große Spanne an Hardware zu unterstützen (Wird von den Linux Distributionen genutzt)
- Erlauben es die Bootzeit zu reduzieren, da die Module später nach dem Booten geladen werden können



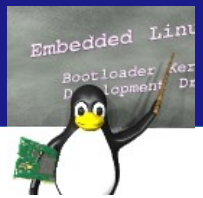
Module Utilities (1)

- `modinfo <module_name>`
  `modinfo <module_path>.ko`
  `Zeigt Informationen zu einem Modul`: Parameter, Lizens, Beschreibung und Abhängigkeiten
- `insmod <module_path>.ko`
  Lädt ein Modul durch Angabe der Kernel Modul Datei



Module Utilities (2)

- `modprobe <module_name>`
  - Lädt ein Modul durch reine Angabe des Modul Namens.
  - Das Modul wird in /lib/modules/<version>/kernel gesucht
  - Benötigte andere Module werden automatisch geladen
- `lsmod`
  - Listet die geladenen Module auf
- `rmmod <module_name>`
  - Entlädt das Modul
  - Wenn das Modul noch in Benutzung ist, schlägt das Entladen fehl
- `modprobe -r <module_name>`
  - Entlädt das Modul und alle davon abhängigen Module



Parameter an Module übergeben

- Verfügbare Parameter herausfinden:
  `modinfo snd-intel8x0m`
- Übergabe mit `insmod`:
  `insmod ./snd-intel8x0m.ko index=-2`
- Übergabe mit `modprobe`:
  Parameter können in `/etc/modprobe.conf` oder in irgendeiner Datei in `/etc/modprobe.d/` eingetragen werden, z.B.:
  `options snd-intel8x0m index=-2`
- Über die Kernel Commandline wenn die Treiber statisch in den Kernel gelinkt sind:

`snd-intel8x0m.index=-2`