

# Linux from Sources

Erik Heinz <erik@42net.de>

23. Oktober 2009

## zur Person

- geboren 1962
- Diplom-Physiker
- Linux seit Januar 1992 (Kernel 0.99pl5)
- 11/1993 Gründungsmitglied des Thüringen-Netz e.V.
- Frühjahr 1994 jengate.thur.de online, erster Internetzugang für Privatpersonen in Thüringen
- 1996 Mitbegründung der IKS GmbH
- seit 07/1998 wissenschaftlicher Mitarbeiter am IPHT

# Warum Quellcodeinstallation?

## Motivation

- Programm nicht als Binärpaket verfügbar
- gewünschte Version, Abhängigkeiten oder compile-time-Konfiguration nicht verfügbar
- Selbstzweck

## Vorteile

- Installation kann individuell angepasst und optimiert werden
- Nachvollziehbarkeit / Reproduzierbarkeit

## Nachteile

- kann aufwändig werden oder fehlschlagen
- Folgeinstallationen können notwendig sein
- Update und Deinstallation nur manuell möglich
- keine automatische Verwaltung der Abhängigkeiten

# UNIX, C und Quellcode

- UNIX-artige Betriebssysteme laufen auf unterschiedlichsten Hardwareplattformen
- Standardisierung und Kompatibilität durch Programmierschnittstellen (System-V, BSD, POSIX)
- Portabilität von Programmen durch Portabilität von (C-)Quellcode
- aber: viele Unterschiede im Detail
- Freie Software als Quellcode (Bsp. BSD, GNU, X11)
- Linux: hohe Verfügbarkeit des Quellcodes, aber Probleme mit divergierenden Versionen und Verfügbarkeit von Informationen

# Installation aus dem Quellcode

- 1 Übersetzen von Quellcode in Objektcode oder anwendungsspezifische Daten
- 2 Kopieren in Systemverzeichnisse
- 3 Konfiguration

## Quellformate

- Programmquelltext: \*.c, \*.cc, \*.S, \*.m4
- Definitionen für Syntax, Parser: \*.l, \*.y
- Internationalisierung (GNU gettext): \*.po
- Dokumentation: \*.tex, \*.xml

## installierbare Formate

- ausführbare Dateien
- ladbarer Objectcode: \*.a, \*.so
- Internationalisierung (GNU gettext): \*.mo
- anwendungsspezifische Daten: sendmail.cf
- Dokumentation: \*.info, \*.html, \*.pdf

# Buildsysteme

## Probleme

- viele Einzelschritte (Bsp. gcc: > 18000 Programmdateien)
- definierte Reihenfolge, Abhängigkeiten
- Portabilität erwünscht

Lösung: spezielles Programmpaket für die Automatisierung des Buildprozesses und Anpassung an lokale Gegebenheiten

## Beispiele

- make
- „Portabilität“ mit `#define` / `#ifdef`
- proprietäre Varianten
- imake
- GNU autoconf/automake
- CMake

# GNU autoconf Schritt für Schritt

- 1 Quellcodepaket beschaffen und auspacken
- 2 Dokumentation lesen (README, INSTALL, `configure --help`)
- 3 Planung (Pfade, Optimierung, Architektur)
- 4 Aufruf `configure`
- 5 ggf. Probleme beheben und zurück zu 4.
- 6 Kontrolle der Makefiles, ggf. Korrektur
- 7 Aufruf `make`
- 8 ggf. Probleme beheben und zurück zu 7. oder 4.
- 9 Aufruf `su; make install`
- 10 Installation kontrollieren und nachbessern, ggf. zurück zu 4.

# Planung der Installation

## Verzeichnisse

- Standards beachten
- private Installationen unter `/usr/local`
- Achtung bei x86\_64-Architektur

## Compileroptionen

- Optimierung
- Architektur, Prozessortyp

## Installationsumfang

- Internationalisierung
- optionale Module
- Dokumentation

# Parameterübergabe an configure und make

- `configure --help`
- Umgebungsvariablen
- notfalls patchen
- Scripte als Automatisierung und Erinnerungshilfe

## Beispiel-Script für configure

```
#!/bin/sh

CFLAGS="-Os -mtune=pentium2 -fomit-frame-pointer" \
CXXFLAGS="-Os -mtune=pentium2 -fomit-frame-pointer" \
LDFLAGS=-s \
configure \
--host=i686-pc-linux-gnu \
--prefix=/usr \
--sysconfdir=/etc \
--mandir=/usr/man \
--infodir=/usr/info \
--localstatedir=/var \
--enable-shared \
--disable-static \
--disable-nls \
--disable-rpath \
--with-readline
```

# Probleme mit configure und make

## Fehlermeldungen finden und verstehen

- `config.log`
- schweigsames `make`
- Ärger mit `libtool`

## dynamische Bibliothek nicht gefunden - was nun?

- nachinstallieren wenn nötig
- `config.log` auswerten
- include-Suchpfad `-I...`
- Linker-Suchpfad `-L...`
- Suchpfad für `pkgconfig`
- Fehler in `configure` oder dem `Makefile`

# Installation kontrollieren und nachbessern

- Prüfen, was installiert wurde und wo
- Liste aufheben für Updates/Deinstallation

## Beispielscript `find_new`

```
#!/bin/sh
MIN=$1
if [ ! "$MIN" ] ; then MIN=60 ; fi
shift
DIRS=$@
if [ ! "$DIRS" ] ; then DIRS="/etc /bin /sbin /lib /usr"; fi
find $DIRS -not -type d \( -cmin -$MIN -or -mmin -$MIN \)
```

## Aufräumaufgaben

- Objektcode strippen
- manpages und info-Files packen
- alte Versionen entfernen
- unbenötigte Teile entfernen (z.B. Dokumentation, Beispiele)

# Links

- Linux From Scratch <http://www.linuxfromscratch.org/>
- FHS <http://www.pathname.com/fhs/>
- Patches Debian <ftp://ftp.debian.org/debian/pool/>
- Patches Redhat <ftp://ftp.redhat.com/pub/redhat/linux/>