Chaosseminar: Linux From Scratch

Nicolas Roeser

nicolas.roeser@ulm.ccc.de

OpenPGP-Key-Fingerprint:
CF93 F253 085A CBA6 2D51
3F59 98CB 2DBE 7DBB CA20

Was ist Linux From Scratch? (1)

- (offiziell) „keine“ Distribution
- Buch

- wpde: „die Anleitung Linux From Scratch“
- wpen: „Linux From Scratch, a kit for building Linux distributions“
- wpen, List_of_Linux_distributions: „A document [...]. (not a distribution).“

Was ist Linux From Scratch? (2)

- „Linux“:
  - Linux-Kernel
  - hauptsächlich GNU-Tools
- „From Scratch“:
  - Beginn beim Stand null: Bootstrapping (Henne-Ei-Problem)
  - nur Quellcode verwendet
  - (plus binäre Pakete zum Bootstrappen)

Hauptunterschiede zu Distros

- keine Binärpakete
- kein Programm für Package-Management: Benutzer ist Paket-Manager
- => keine automatische Auflösung von Abhängigkeiten bei Installation weiterer Pakete
- höhere Optimierung/Anpassung an Prozessor+Architektur möglich (Gentoo)

Warum LFS?

- Unzufrieden mit Distributionen? ;-)
- Volle Kontrolle über das eigene System
- Was dazu lernen
- Blick „unter die Haube“
- Wenig Platz auf dem Zielmedium
- Sicherheitsüberlegungen (dann muss man die Sources aber auch lesen!)
- Optimierung der Bibliotheken und Programme

Wer benutzt LFS?

- Mehr als 16.000 registrierte (und viel mehr unregistrierte) Benutzer
- verschiedene Bereiche:
  - Desktop
  - Embedded
  - Roboter
  - Server
  - spezielle Abkömmlinge (z. B. LSB-Build)

Wie installiert man Software?

- Software ist in Paketen organisiert
- Oft größere Unterschiede zwischen Dateien in Source-Paketen und Binär-Paketen (mehr Binärpakete, z. B. *-dev)

- `configure`
- `make`
- `make check`
- `make DESTDIR="${WHEREEVER}" install`
- `$WHEREEVER/ anschauen`
- `make install`

Pakete ändern/reparieren

- Quellcode editieren
- Oft Editor (z. B. vim) und ggf. anschließend autotools

- In LFS:
  - patch
  - sed
  - normalerweise keine Änderung von autotools-Quelldateien, sondern stattdessen von Makefile.in etc.

Toolchain (Überblick)

- glibc: C-Bibliothek
- gcc: C-Compiler
- binutils: Assembler, Linker

- C-Compiler ruft Assembler auf
- Linker bindet Programme:
  - statisches Linken
  - dynamisches Linken: shared Libraries

- wichtiges „Problem“: NSS-Libraries

Geschichte von LFS (1)

LFS HOWTO 1.0 (Dezember 1999):

- HOWTO@linuxdoc von Gerard Beekmans
- Pakete:
  - „Basissystem“ (Toolchain und Hilfsprogramme)
  - glibc-2.0.7pre6 (statt glibc-2.1.2: kein gcc-2.8)
  - gcc-2.95.2 und gcc-2.7.2.3
  - LILO
  - telnet, lynx
  - Mail-System (fetchmail, sendmail, procmail, mailx, mutt)
  - Apache httpd
  - Xfree86, WindowMaker
  - rc-Skript und dpkg aus Debian

Geschichte von LFS (2)

LFS HOWTO 1.0 (Dezember 1999):

- für x86 geschrieben
- Alles wird als root erledigt
- u. a. /usr/local -> . (inkompatibel zu FHS)
- /dev, /etc/{passwd,group} etc. kopiert vom Host-System
- Ziel: lauffähige glibc
  - einige Pakete erst statisch gelinkt
  - make vom Host-System aus, Installation in neuem System (Reboots)
- Patchen mit Editor
- nur für erfahrene Benutzer geeignet

Geschichte von LFS (3)

- LFS HOWTO 1.1 und 1.2: kleine Updates
- LFS HOWTO 1.3:
  - Community: Mailingliste(n)
  - spanische Übersetzung begonnen
  - Unterstützung für shadowed passwords
  - „fixed packages“ (Pakete mit LFS-Patches)
  - Debian-Binärpakete für libstdc++-2.91.60 (da Sources unauffindbar)
  - Mail-Software, XFree86 etc. optional

Geschichte von LFS (4)

- LFS HOWTO 2.0-beta2 (ca. Februar 2000):
  - viele Erklärungen:
    - tar
    - gzip
    - $LFS
  - glibc-Installation ohne Reboot
- LFS HOWTO 2.1-pre1: FHS-kompatibel
- LFS HOWTO 2.2 (März 2000):
  - linuxfromscratch.org
  - „this HOWTO“ -> „this document“

Geschichte von LFS (5)

- LFS 2.3.1 (April 2000):
  - „LFS book“
  - zusätzliche Mailingliste „linux“ (für OT-Sachen)
  - OASIS DocBook SGML 3.1, mehrere Dateien
  - Informationen für Apple PowerPC
  - wesentlich übersichtlicher als zuvor
- LFS 2.3.2 (April/Mai 2000):
  - Web Mirrors
  - PowerPC in eigenen Kap., optionale Kap. raus
  - Editor ed
  - NSS-Bibliotheken von Host-System kopiert
  - MAKEDEV und devfs erwähnt (nicht benutzt)
  - chroot nach Bauen des statischen Build-Systems

chroot (allgemein)

- Konzept zum Wechsel des Wurzelverzeichnisses („root directory“, „/“)
- chroot(2): BSD-, X/OPEN-Systemcall
- chroot(1): Programm, das ihn ausführt

- Benutzt z. B. zum Schutz des Systems vor bösen Mädels und Jungs, die sich Zugang via Internet-Dienste verschaffen können

chroot (LFS)

- Zur „Umschaltung“ auf gebaute Tools
- Fällt schon jemandem das Problem auf ...?
- Tipp: hardgecodete Pfade zu Binaries (Programmen/Bibliotheken)
- Immer noch keine Idee? Abwarten ...

Zwischenstand: Die Methode von LFS 2.3.2

- Vereinfachte Zusammenfassung
- Host-System:
  - configure --prefix=/usr
  - make LDFLAGS=-static
  - make prefix=$LFS/usr install
- chroot
- chroot-System:
  - configure --prefix=/usr
  - make
  - make install

Geschichte von LFS (6)

- LFS 2.3.3 (Mai 2000):
  - Buch-Quelldateien reorganisiert, übersichtlicher
  - neues rc-Skript
- LFS 2.3.4 (Juni 2000):
  - neue Mailinglisten:
    - lfs-apps: Support für Programme außerhalb LFS
    - lfs-config: Konfiguration von LFS-Software
- LFS 2.3.5 (Juni 2000):
  - MAKEDEV
  - Runlevels gemäß LSB-Empfehlung
  - Optimierung (O3, mcpu, march in CFLAGS)
  - ALFS begonnen

Automated Linux From Scratch

- Automatisierung von LFS-Builds
- Eliminierung von Fehlern bei Copy-and-Paste
- => Effizienzsteigerung

- Zusätzliches Projekt, Benutzung freiwillig. LFS bleibt in Buchform und Standard-Weg für Builds

Geschichte von LFS (7)

- LFS 2.3.6 (Juli 2000):
  - Probleme mit Optimierung tauchten auf; Hinweis
- LFS 2.3.7 (August 2000):
  - Umstellung von DocBook SGML auf DocBook XML 4.1.2
- LFS 2.4 (August 2000):
  - Quellen-Baum mal wieder reorganisiert:
  - zwei Bücher: PowerPC + x86
- LFS 2.4.1 (Oktober 2000):
  - Lizenzänderung: LDP -> BSD-ähnlich
  - seltsame Experimente (z. B. vim als erstes)

Geschichte von LFS (8)

- ...
- LFS 3.0 (September 2001):
  - größeres Update der Projekt-Infrastruktur
  - „richtiges“ ChangeLog
  - heute bekannte Ch5/Ch6-Aufteilung
  - nur noch XML (+DSSSL)
  - Abgeschwächte advertising clause in Lizenz
  - Optimierungs-Warnung für Toolchain-Pakete
  - Probleme mit chroot-Ansatz bei Wechsel von altem Kernel auf linux-2.4 bemerkt, Warten auf Bugreports

Geschichte von LFS (9)

- ...
- LFS 3.2-rc1 (Februar 2002):
  - Build als Benutzer „lfs“ statt als root zum Bauen des statisch gelinkten Systems

„Reboot into static/chroot“

- Ansatz von einigen LFSern:
- Software statisch gelinkt bauen wie bisher
- neuen Kernel kompilieren (unkritisch, da unabhängig von glibc)
- zusätzliche benötigte/gewünschte Software statisch gelinkt bauen
- kein chroot, statt chroot Reboot ausführen

- Fix für Problem vom 2002-08-18 (von linux-2.2.x zu LFS CVS)
  - Beschreibung vermutlich falsch, Problem real

Geschichte von LFS (9)

- LFS 4.0-rc1 (September 2002):
  - keep_chap5_and_chap6_sep.txt (Hint) integriert:
  - /static für statisch gelinkte Software in neuem System
  - Rest wie üblich installiert
  - /static kann am Schluss entfernt werden, da nicht mehr benötigt
  - saubererer Ansatz

PLFS – „Pure“ LFS (1)

- Kommentar von Kernel-Entwickler Alan Cox zu LFS auf linux-kernel (2002-10-27):
  „I get so many weird never duplicated reports from linux from scratch people that don't happen to anyone else that I treat them with deep suspicion. Especially because it sometimes goes away if they instead build the same kernel with Debian/Red Hat/.. binutils/gcc“
- Das musste sich ändern ...

PLFS – „Pure“ LFS (2)

- LFS-Hacker um Greg Schafer & Ryan Oliver kümmern sich ab ca. Januar 2003 um das Problem
- Ziel: „Purity“, also Reinheit des neuen Systems
- Beweis über ICA: Iterative Comparison Analysis; Schwierigkeiten: Datum etc.
- Automatischer Build über Skripte
- Entwicklung parallel zu „mainstream“ LFS
- make check!
- schließlich Integration in LFS (ab Version 5)

Toolchain und ld.so

- Toolchain:
  - binutils
  - gcc
  - glibc
- Dynamischer Linker: ld.so:
  - /lib/ld-linux.so.2
  - Programm
  - lädt dynamische Bibliotheken

Wie funktioniert LFS heute?

- Stand: Version 6.1.1 (30. November 2005)

- Ch5 in /tools
- libc-headers
- „Locking in glibc“
- chroot
- Ch6
- FHS: /media und /srv
- gcc3 auch f. Kernel (linux-2.6). GRUB, udev.
- Warnung vor Optimierung (aber viele tun es trotzdem ;-))
- einige Übersetzungen

Voraussetzungen

- Laufendes System mit Linux-Kernel und einigen Programmen (z. B. gcc)

  (für die Geeks: irgendein Kernel unter UNIX-ähnlichem System mit halbwegs ANSI C-kompatiblen Compiler genügt auch)

- Freie Partition

  (für die Geeks: ausreichend Ahnung genügt auch. Installation neben Deb^H^H^H^H anderer Distro möglich. :-p)

Ch5: Building the tools (1)

- Voraussetzung: Host-System hat Linux 2.6, kompiliert mit gcc3 (kein Problem)
- Ziel: Erzeugen von Tools, die
  - vom Host-System unabhängig sind,
  - zum Kompilieren eines neuen Systems verwendet werden können.
- Bekannte Analogie: Bootstrappen von gcc (bootstrap, profiledbootstrap (bei gcc-4), bootstrap3, bootstrap4)

Ch5: Building the tools (2)

- Symlink /tools -> ${LFS}/tools; in $PATH
- binutils (gegen Host-glibc) in /tools installieren
- ld vorbereiten für später, Suchpfad /tools/lib
- gcc in /tools installieren (bootstrap ohne fixincludes).
  Achtung, „GNU magic“: gcc sucht binutils erst in $prefix und nicht in $PATH!
- libc-headers und glibc in /tools installieren
- Wir haben jetzt temporäre libc;
  gcc benutzt (noch) /lib/ld-linux.so.2

specs

- gcc benutzt /usr/lib/gcc/.../.../specs
- Was steht da drin?

- Linker ändern
- gcc benutzt jetzt /tools/lib/ld-linux.so.2 und schreibt den als dynamischen Linker in erzeugte Programme. Toll!

„Locking in“ glibc

- Linker-Skripte anpassen (SEARCH_DIR)
- specs-Datei von gcc anpassen

- Wir haben jetzt vom Host abhängige Toolchain, die glibc in /tools für neu kompilierte Programme verwendet

Ch5: Building the tools (3)

- gcc in /tools installieren (benutzt jetzt neuen Linker und neue glibc!)
- binutils (gegen neue glibc) in /tools installieren

- Wir haben jetzt neue, vom Host unabhängige Toolchain in /tools,
- neu kompilierte Programme verwenden /tools/lib/ld-linux.so.2 und glibc in /tools,
- also nicht mehr aus dem Host-System.

Ch5: Building the tools (4)

- Restliche Software installieren wie gehabt
- Aber: wir linken die Programme jetzt dynamisch gegen (die neue) glibc!

Ch6: Essential system software

- chroot
- libc-headers und glibc neu installieren (zum zweiten und letzten Mal): mit neuer Toolchain gebaut :-)
- Toolchain „readjusten“! => Linken gegen Bibliotheken in /lib, /usr/lib (nicht /tools/*)
- binutils neu installieren (mit neuer Toolchain)
- gcc und restliche Software neu installieren (Software wird dynamisch gegen installierte glibc gelinkt)
- Rest im Prinzip wie gehabt

Diverses

- SBU
- tooldir bei binutils
- fixincludes von gcc
- Linux kernel headers & why not to use them
- FBBG
- Re-LFS
- ADA-Compiler
- glibc bei FBBG: -O2 (-O0 geht nicht)
- Updaten von glibc

Heiße Tipps für den Schluss

- Am Ende nicht gleich das neue LFS booten, sondern erst mal GPM und einen Textbrowser installieren
- Nie als root kompilieren etc.
- Paket kaputt? Patch upstream senden!
- DVD-Brenner besorgen: Sources sind groß
- Nicht automatisch Sachen aus SVN-/CVS-Repositories übersetzen und installieren lassen (if you don't know what you're doing)

Zukunft von LFS

- Verbesserungen im Bereich udev/hotplug
- LFS Multiarch
- mehr HLFS?
- (alphabetische Build-Reihenfolge)
- „More Control and Package Management [...]“
- cross-lfs (ehemals lfs-hackers, PLFS, keep_chap5_and_chap6_sep)
- vollständige UTF-8-Unterstützung

LFS und Subprojekte

- LFS
- HLFS
- Hints
- BLFS (inkl. Java und OpenOffice)
- ALFS
- (FAQ)
- Patches
- LiveCD

Ende

- Fragen?
- Parasco