Christoph Junghans, Andreas K. Hüttel, Ulrich Müller

Maßarbeit

Gentoo Linux: Quelltexte und Rolling Releases

Gentoo ist die etwas andere Linux-Distribution: Anwender installieren Softwarepakete aus den Quelltexten, die Programme werden permanent auf dem aktuellen Stand gehalten. Dank einer ausgefuchsten Paketverwaltung ist das erstaunlich einfach.

Als Daniel Robbins 1999 Gentoo Linux startete, war sein Ziel ursprünglich eine Meta-Distribution – ein Linux-System, das seine Pakete im Quelltext bereitstellt und als Grundlage zum Bau eigener Distributionen dient. Mittlerweile arbeiten etwa 150 Entwickler an dem System, das rund 10 000 Programmpakete in seinen Repositories vorhält – und einen näheren Blick für jeden lohnt, der sich ein individuelles Linux-System selbst maßschneidern will.

In der Gentoo-Welt werden Programme bei der Installation eines Programmpakets auf dem Rechner des Benutzers kompiliert, wofür nur der Quellcode und eine Bauanleitung – das so genannte Ebuild – erforderlich sind. Dieser Ansatz gibt dem Benutzer viele Möglichkeiten zur Feinjustierung in die Hand, die er bei den Binärpakten der traditionellen Distributionen nicht hat. So ist es möglich, paketweise oder global Unterstützung für bestimmte Funktionen wie X11 oder Python, KDE oder Gnome ein- oder auszuschalten, je nachdem, ob am Ende ein Desktop, ein Server oder ein minimales Rettungssystem stehen soll.

Auch die Interessen des Benutzers spielen eine wichtige Rolle: Auf einem KDE-System soll die Versionsverwaltung Subversion Passwörter vielleicht im KDE-Passwortsafe KWallet speichern, auf einem anderen Rechner im Gnome-Schlüsselring oder ganz ohne GUI. Ein Perl-Programmierer benötigt im Texteditor seiner Wahl Syntax-Highlighting für Perl, für andere Anwender ist das überflüssig und Platzverschwendung.

Größter Nachteil der Quelltextpakete ist der Aufwand beim Kompilieren – die Installation großer Pakete wie KDE erfordert Geduld. Andererseits lässt sich so ein auf den jeweiligen Einsatz optimiertes System einrichten. Zudem ist es relativ einfach, aus dem Quellcode eine Installation für ungewöhnliche oder neue Hardware zu erzeugen – Gentoo unterstützt mittlerweile über zehn verschiedene Prozessor- und Rechnerarchitekturen, von Embedded ARM bis hin zum S/390-Mainframe.

Software installieren …

Die Installation von Gentoo ist aufwendiger als bei den meisten anderen Distributionen, da viele Entscheidungen dem Benutzer selbst überlassen werden. Das Gentoo-Handbuch liefert hier jedoch ausführliche und verständliche Anweisungen [1]. Ist Gentoo installiert, werden aktualisierte Pakete als „Rolling Release“ eingespielt, sobald sie verfügbar sind – eine Installation sollte so für die restliche Lebenszeit des Rechners halten.

Updates spielt man mit dem Paketmanager Portage ein, der sich um die Auflösung von Abhängigkeiten und das Kompilieren der Pakete kümmert – man muss sich nicht so tief mit den Interna beschäftigen wie etwa bei Linux from Scratch [2]. Um beispielsweise den Editor vim zu installieren, ruft man einfach das Portage-Frontend emerge auf:

```
emerge --ask --verbose vim
```

Emerge gibt die Abhängigkeiten von vim aus (das Paket vim-core) sowie die standardmäßig verwendeten USE-Flags, die die Unterstützung bestimmter Funktionen und die Installation optionaler Programmbausteine kontrollieren:

```
USE="X acl cscope gpm nls perl ruby -debug
     -minimal -python -vim-pager"
```

Mit diesen Einstellungen wird vim mit Unterstützung unter anderem für X11 und ACLs gebaut. Der Editor ist mit Perl und Ruby skriptbar, der eingebaute Python-Interpreter ist jedoch durch `-python` deaktiviert.

Ein kleines Paket von Skripten, gentoolkit, das auf fast jedem Gentoo-Rechner installiert ist, hilft bei der Paketverwaltung. So erklärt der Befehl

```
euse -i gpm
```

die Bedeutung des USE-Flags „gpm“, das die Mausunterstützung auf der Kommandozeile steuert. `euse -E` und `euse -D` schalten Flags systemweit an und ab; auf Paketebene empfiehlt sich dafür das Programm flaggie:

```
flaggie app-editors/vim -X
```

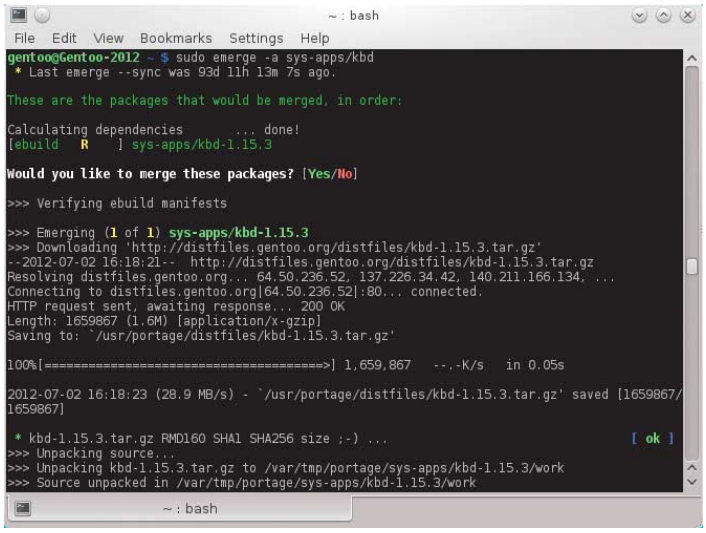
schaltet die X11-Unterstützung im Editor vim ab.

… und optimieren

Alle Einstellungen sind in in editierbaren Textdateien unter /etc gespeichert: globale USE-Flags und Portage-Einstellungen wie Compiler-Optionen, Pfade und die verwendeten Quelltext-Mirrors in /etc/make.conf, paketspezifische USE-Flag-Änderungen in /etc/portage/package.use.

Neue Pakete werden bei der Installation standardmäßig mit dem Gnu C Compiler gcc und einem Satz von Optimierungsoptionen übersetzt, die der Benutzer in /etc/make.conf festlegt. Normalerweise empfehlen sich einfache Einstellungen wie `-march=native -O2 -pipe` – das erzeugt Binärcode für den vorhandenen Prozessor und führt eine Optimierung der Stufe 2 durch. Auf Embedded Systems würde man eher `-Os` für Größenoptimierung verwenden, rechenintensive Simulationen können von aggressiveren Optimierungen wie `-O3` profitieren.

Alle diese Einstellungen können systemweit, paketweise oder für Paketkategorien festgelegt werden – Details dazu und zu den



Emerge ist das Standard-Werkzeug zur Interaktion mit der Paketverwaltung.

eshowkw zeigt an, welche Versionen für welche Architekturen verfügbar sind.

möglichen GCC-Optionen finden Sie über den c't-Link am Ende des Artikels. Alternative Compiler wie clang oder Intels icc sind in der Erprobung, doch es lassen sich noch nicht alle Pakete mit ihnen übersetzen. Auch kommerzielle Programme wie der Adobe Reader lassen sich installieren, indem der Benutzer die Installationspakete des Herstellers selbst herunterlädt und Gentoo nur das Ebuild bereitstellt, der die Dateien in das System integriert.

Außer für die Paketinstallation ist emerge auch für Deinstallationen zuständig (Option `--unmerge`), kann Pakete suchen (`--search`) und detaillierte Informationen über ein Paket ausgeben (`--info`). Man kann die zu installierenden Dateien nach dem Kompilieren auch in Archiven speichern (`--buildpkg`), um sich bei einer erneuten Installation oder Installationen auf anderen Rechnern das Kompilieren zu sparen (`--usepkg`). Diese Archive entsprechen den Binärpaketen anderer Distributionen.

Wem die Kommandozeile nicht so liegt, der kann das Portage-GUI Porthole verwenden. Für spezielle Aufgaben empfiehlt sich equery aus dem bereits erwähnten Paket gentoolkit, das Informationen über die Größe von Paketen und die zugehörigen Dateien, die Abhängigkeiten und die Versionsgeschichte von Paketen aus der Portage-Datenbank ausgibt. Zur bequemeren Paketsuche gibt es das Tool eix, das über eine eigene Datenbank verfügt und so die Suche beschleunigt. Portage kommt nicht nur in Gentoo zum Einsatz: Google verwendet die Paketverwaltung in seinem Chrome OS.

Versionen

Jedes Paket kann in verschiedenen Versionen vorliegen. Normalerweise installiert emerge Pakete, die als „stable“ markiert sind – diese Pakete sind gut getestet und arbeiten mit den restlichen stabilen Paketen zusammen. Das Keyword „testing“ bezeichnet Pakete, die zwar getestet wurden, aber noch nicht als stabil eingestuft sind – unter Umständen sind diese Pakete auch nur zu neu, sodass noch nicht genügend Erfahrungsberichte vorliegen. Einen Überblick über die vorhandenen Versionen eines Pakets erhält man mit dem Tool eshowkw aus dem Paket gentoolkit. Die Ausgabe für Cups sehen Sie in der Grafik oben rechts.

Hier steht jede Zeile für eine Paketversion und jede Spalte für eine Rechnerarchitektur: cups-1.5.2-r4 beispielsweise ist stabil für AMD64 (+), aber noch testing (~) für S390. Installiert ([I] am Anfang der Zeile) ist die Version 1.5.3; die Betaversion 1.6_beta1 ist noch gar nicht klassifiziert (o steht für leer). [M] kennzeichnet maskierte Pakete, die noch getestet werden müssen, schwerwiegende Probleme aufweisen (Instabilität, Inkompatibilität, kritische Sicherheitslücken) oder demnächst aus der Distribution entfernt werden – im Beispiel die Version 9999 des Pakets, ein sogenannter „live ebuild“, der den jüngsten Entwicklungsstand von Cups direkt aus dessen Versionsverwaltung installiert. Das Paket ist maskiert, da es keine Garantie gibt, dass das Kompilat überhaupt funktioniert. Versucht man, ein maskiertes Paket zu installieren, meldet der Paketmanager den Grund für die Maskierung zurück.

„testing“-Pakete muss man explizit erlauben, im Beispiel vim auf der AMD64-Architektur:

```
flaggie app-editors/vim +~amd64
```

Man kann eine Installation auch komplett aus „testing“-Paketen aufbauen, indem man die Zeile

```
ACCEPT_KEYWORDS="~amd64"
```

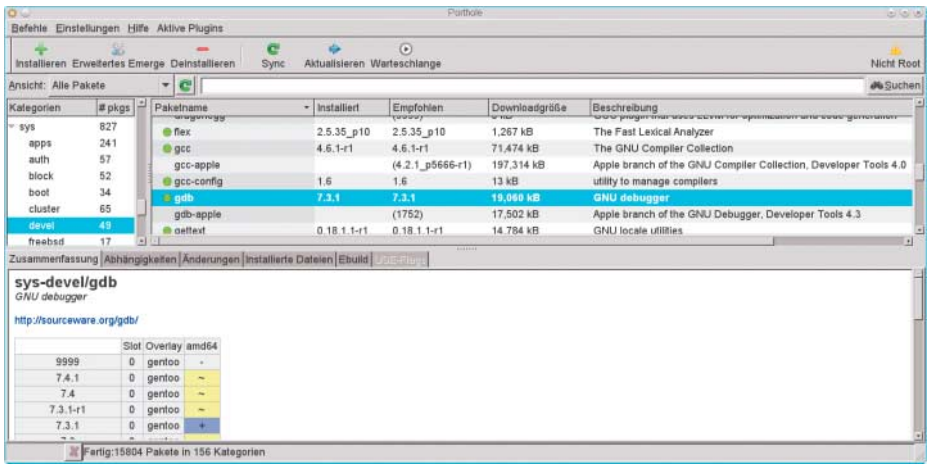
in die Datei /etc/make.conf einfügt – allerdings werden dann häufig wenig getestete Programmversionen eingespielt. Maskierte Pakete sind noch problematischer; will oder muss man solch ein Paket trotzdem installieren, kann man es demaskieren, indem man es in /etc/portage/package.unmask einträgt. Für die neueste Version von gcc lautet die Zeile

```
=sys-devel/gcc-4.7.1
```

Einen guten Überblick über alle Pakete im offiziellen Paketbaum findet man unter [3].

Bauen

Sobald Portage die Abhängigkeiten der Pakete anhand der Informationen in seiner Paketdatenbank berechnet hat, führt es für jedes zu installierende Paket das Ebuild aus, ein bash-ähnliches Skript, das den Kompilier- und Installationsprozess steuert. Sein Inhalt ist dem manuellen Vorgehen bei der Installation des Quelltextpakets – häufig `./configure`



Porthole ist ein GUI für das Portage-System.

&& make && make install – sehr ähnlich. Ein Ebuild ist nicht schwer zu schreiben, wenn das Paket ein Buildsystem wie GNU Autotools oder CMake verwendet; das Ebuild des Editors Zile beispielsweise enthält nur einige Zeilen (siehe Listing unten).

Portage ruft während der Installation mehrere Phasen auf, die nicht alle explizit im Ebuild vorhanden sein müssen; so verwendet Zile die Standardfunktion des Paketmanagers für „src_unpack“, die einfach die Quellen aus dem in SRC_URI genannten tar-Paket auspackt.

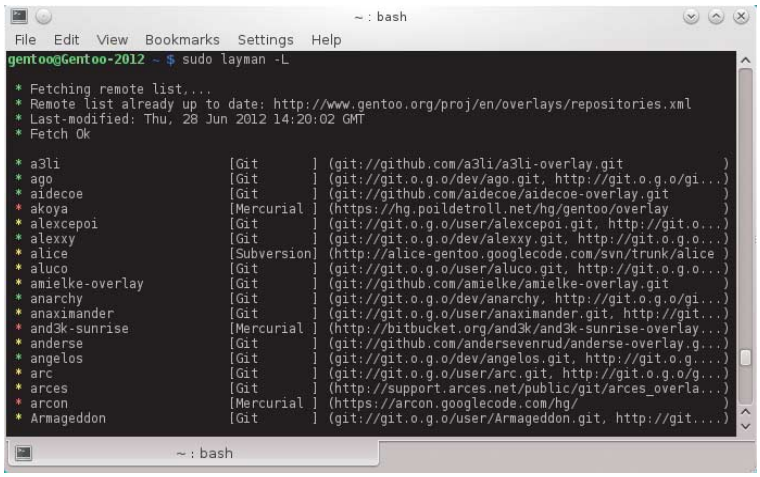
In der Phase „src_configure“ werden die USE-Flags in Optionen des configure-Skripts des Pakets übertragen, wobei die Funktion use_enable benutzt wird, die --enable-XX oder --disable-XX für das ein- oder ausgeschaltete USE-Flag XX zurückgibt. Die Übergabe anderer Optionen ist selbstverständlich ebenfalls möglich, wie im Beispiel --docdir für den Installationsort der Dokumentation.

Die Phase „src_compile“ ist im Zile-Ebuild nicht definiert – es kommt die Standardfunktion zum Einsatz, die im wesentlichen make aufruft. In src_install wird das Paket schließlich installiert, zunächst jedoch nicht am endgültigen Ort, sondern in einer „Sandbox“ im Verzeichnis ${ED}. Der letzte Schritt ist das Verschieben des Pakets von ${ED} nach ${ROOT}, was Portage wieder ohne Zutun des Ebuilds erledigt. Dabei verzeichnet die Paketverwaltung alle Dateien des Pakets in ihrer Datenbank, um sie bei einer späteren Deinstallation oder bei einem Update wieder entfernen zu können.

Eclasses, das sind Bibliotheken von Funktionen für Ebuilds, erlauben es, einheitliche und einfache Ebuilds zu schreiben und Code-Duplikation zu vermeiden. Die autotools-utils stellen Funktionen für die vereinfachte Nutzung von GNU Autotools bereit, cmake-utils bietet ähnliche Funktionen für cmake-basierte Buildsysteme. Für Details und kompliziertere Fälle liefert das Entwicklerhandbuch nützliche Hinweise, das Sie über den c’t-Link finden.

```
EAPI=4
DESCRIPTION="Zile is a small Emacs clone"
HOMEPAGE="http://www.gnu.org/software/zile/"
SRC_URI="mirror://gnu/zile/${P}.tar.gz"
LICENSE="GPL-3"
SLOT="0"
KEYWORDS="~alpha amd64 ppc ~sparc x86 ~
          ~x86-freebsd ~amd64-linux ~x86-linux ~
             ~ppc-macos ~x86-macos ~x86-solaris"
IUSE="acl test"
RDEPEND="dev-libs/boehm-gc
  sys-libs/ncurses
  acl? ( virtual/acl )"
DEPEND="${RDEPEND}
  test? ( dev-lang/perl )"
src_configure() {
  econf \
    --docdir="${EPREFIX}"/usr/share/doc/${PF} \
    --disable-silent-rules \
    $(use_enable acl)
}
src_install() {
  emake DESTDIR="${D}" install
  dodoc README THANKS
  # Zile should never install charset.alias
  rm -f "${ED}"/usr/lib/charset.alias
}
```

Das Ebuild-Skript steuert den kompletten Übersetzungsprozess.



Viele weitere Programme für Gentoo sind als Overlays erhältlich.

Overlays

Da es sehr einfach ist, neue Ebuilds zu schreiben, steuern inzwischen viele Benutzer neue Pakete über den Gentoo-Bugtracker bei oder stellen sie in so genannten Overlays bereit, die eine neue oder modifizierte Paketversion über den Standard-Paketbaum legen. Es kann verschiedene Gründe haben, dass ein Paket als Overlay bereitsteht – das Ebuild genügt den Qualitätsansprüchen der Gentoo-Entwickler nicht, es handelt sich um eine Alpha- oder Betaversion, oder eine Gruppe von Benutzern betreut eigenständig eine Sammlung von Programmpaketen. Man kann ein Overlay direkt herunterladen und über einen Eintrag in /etc/make.conf in das System integrieren, komfortabler geht es jedoch mit dem Programm layman. layman -L liefert eine Liste aller bei Gentoo registrierten Overlays. In den Weiten des Internets existieren zahlreiche weitere Overlays, die man über eine Suchmaschine finden kann [4].

Von Gentoo abgeleitete Distributionen wie Funtoo oder Sabayon benutzen häufig Overlays, um ihre Eigenheiten ins System zu bringen – Gentoo erfüllt hier aufgrund seiner einfachen Erweiterbarkeit seine ursprüngliche Aufgabe einer Meta-Distribution als Basis für Anpassungen und Erweiterungen. Sabayon verwendet vorkompilierte Pakete, die es über den eigenen Paketmanager entropy bereitstellt; das reduziert die Installationszeit, allerdings auf Kosten der Flexibilität. Auch für Gentoo stehen vorkompilierte Pakete bereit (siehe c’t-Link).

Wenn man ein Paket in der gewünschten Version nicht finden kann und kein Ebuild schreiben will, kann man das Standardpaket einfach um einen Patch erweitern. Dazu legt man die Patch-Dateien vor der Installation des Pakets category/package im Verzeichnis /etc/portage/patches/category/package ab. Vor dem Kompilieren werden sie auf den Quelltext angewendet. Diese Funktion ist auch zur Softwareentwicklung und zum Debuggen hilfreich.

Die Flexibilität von Gentoo macht Dinge möglich, die für Linux-Distributionen untypisch sind. So kann der Linux-Kernel durch einen FreeBSD-Kernel ersetzt und sogar ganz weggelassen werden: Die Gentoo-Pakete können unter anderen Unix-Systemen wie Mac OS X, aber auch unter dem Unix-Subsystem Interix für Windows einfach in ein beliebiges Unterverzeichnis installiert werden. Portage erzeugt und installiert dann auf das Gastgeber-System angepasste Programme, was mit emerge viel einfacher geht als von Hand. Solch ein so genanntes „Prefix-System“ ist nützlich, wenn man auf einem System keine Adminrechte hat und daher keine Software systemweit installieren darf, aber eine Vielzahl von Programmen braucht.

Erheblich zur Popularität von Gentoo trägt bei, dass die Distribution aufgrund der Kombination aus quellenbasierter Verteilung und „rolling release“, also dem kontinuierlichen Update der Pakte, neue Programmversionen (zumindest als „testing“) sehr schnell bereitstellt. Ebuilds für neue Versionen von KDE und LibreOffice existieren beispielsweise fast immer schon am Tag der Freigabe der neuen Version. In der Regel ist es kein Problem, einzelne „testing“-Pakete mit einem stabilen Grundsystem zu kombinieren.

Das macht Gentoo sicher nicht zum idealen Linux für Einsteiger, aber fortgeschrittene Benutzer können sich so von den Release-Zyklen und Einschränkungen im Funktionsumfang der großen Distributionen unabhängig machen. Die Kombination aus USE-Flags und dem Paketmanager Portage machen es dem Anwender einfach, sich eine optimale Linux-Distribution für nahezu jeden Zweck zu bauen. (odi)

Literatur

[1] Gentoo-Handbuch: www.gentoo.org/doc/en/handbook/
[2] Oliver Diedrich, Selbst gebaut, Mit Linux From Scratch zur eigenen Distribution, c’t 22/02, S. 214
[3] Überblick über die Pakete: http://packages.gentoo.org/
[4] Suchmaschine für Overlays: http://gpo.zugaina.org/

www.ct.de/1216162 c't