Gunnar Wrobel

Gentoo Linux

Installation – Konfiguration – Administration

Bibliografische Information Der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet die Originalausgabe dieser Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Ausgabe ist textidentisch mit der Originalausgabe:
Open Source Press, München 2008 [ISBN 978-3-937514-34-5]
Gesamtlektorat: Dr. Markus Wirtz
Satz: Open Source Press (LaTeX)

Inhaltsverzeichnis

Vorwort

Ein wichtiger Hinweis gleich vorweg: Dieses Buch ist nicht für den Linux-Anfänger geschrieben. Zum einen ist Gentoo nicht gerade eine typische Einsteiger-Distribution, und zum anderen würden die Beschreibungen ausufern, wenn wir Leser ansprechen wollten, denen der Umgang mit der Kommandozeile nicht vertraut ist. Wer aber ein wenig Linux-Erfahrung mitbringt, keine Berührungsängste hat und Gentoo noch nicht kennt, dem bietet dieses Buch einen sauberen Einstieg. Darüber hinaus möchte es für den Gentoo-Kenner ein hilfreiches Nachschlagewerk bei der täglichen Arbeit mit dem System sein.

Ohne Gentoo-Kenntnisse geht es zunächst einmal darum, das System zu installieren und damit eine Grundlage zu schaffen, auf der man experimentieren und arbeiten kann. Entsprechend starten wir in Kapitel 1 mit der grundlegenden Installation. Dabei haben wir darauf geachtet, dass dieses Buch und die beiliegende DVD zusammen mit einem geeigneten Rechner und ein wenig Festplattenspeicher ausreichen, um das System erfolgreich zu installieren. Eine Netzwerkanbindung ist nicht notwendig.

Wer die Distribution schon kennt wird im Installationskapitel in groben Zügen die Instruktionen des Gentoo-Installationshandbuches wiederfinden, aber an vielen Stellen gehen wir darüber hinaus, um den Installationsprozess zu veranschaulichen und damit ein Grundverständnis für das Gesamtsystem zu schaffen.

Sobald das System steht, gehen wir in den Kapiteln 2 bis 10 schrittweise auf die wichtigsten Konfigurationsoptionen und Portage, das Paketmanagementsystem von Gentoo, ein. Wir legen hier zwar noch das Augenmerk auf den Gentoo-Neuling, doch können einige Abschnitte nach der Erstinstallation zunächst einmal übersprungen werden bzw. sind eher für erfahrene Anwender von Interesse. Gerade im Zusammenhang mit der Konfiguration sind hier aber tiefergehende Informationen sinnvoll und als Referenz zum späteren Nachschlagen zu nutzen.

Ab Kapitel 11 lassen wir die Installation hinter uns und beschäftigen uns mit fortgeschrittenen Themen für die langfristige Pflege, Nutzung und Erweiterung des Systems. Hier finden Sie Informationen zu Programmen und Optionen, die ich als langjähriger Gentoo-Nutzer und mittlerweile aktiver

Entwickler immer wieder aus verschiedensten Quellen nachschlagen musste, so dass ich glaube, dass auch andere erfahrene Anwender einen Nutzen aus dieser Zusammenstellung ziehen werden.

Sollte damit die Zielsetzung des Buches umrissen sein, so bleibt in dieser Einleitung noch die möglicherweise wichtigste Frage zu klären ...

Warum Gentoo?

Das Gentoo-Projekt hat seine Distribution nach dem schnellsten Schwimmer unter den Pinguinen benannt, dem Gentoo, und es suggeriert damit, ein mit Gentoo Linux betriebenes System in Bezug auf die Geschwindigkeit der verwendeten Software maximal optimieren zu können. Das ist sicherlich nicht ganz falsch, denn eine Distribution, bei der man sämtliche Software selbst aus dem Source-Code kompiliert, bietet mehr Einflussmöglichkeiten als eine aus vorgefertigten Binärpaketen bestehende, da sie es gestattet, den Kompiliervorgang für die Zielhardware zu optimieren.

Wen jedoch die Hoffnung auf ein merklich schnelleres Linux-System zur Installation von Gentoo treibt, mag zur Überzeugung kommen, dass der deutsche Name desselben Vogels die Sache besser trifft: Eselspinguin ... Den Geschwindigkeitsgewinn optimal auf den Prozessor abgestimmter Software gegenüber generisch für i386 gebauten Paketen nimmt der Anwender im Betrieb kaum wahr. Dafür schlägt die Zeit, die man bei Gentoo für die Kompilation der Software benötigt, durchaus zu Buche; sie summiert sich zu einigen Stunden Rechenzeit, und so fragt man sich schnell, ob man da nicht an der Nase herumgeführt wurde.

So liegt der Vorteil eines Gentoo-Systems trotz des Namens nur in geringem Maße in der Geschwindigkeit. Es ist zwar richtig, dass sich die installierte Software über entsprechende Compiler-Flags in ihrer Geschwindigkeit optimieren lässt. Dieser Vorgang ist jedoch keinesfalls simpel und treibt selbst Linux-Profis hin und wieder zur Verzweiflung, weil die Kombination mancher Optionen bei bestimmten Softwarepaketen zu schwer identifizierbaren Fehlern führt. Gerade Anfänger gehen hier schnell in eine Falle und wenden sich irritiert von der Distribution ab.

Die Vorteile

Wirklich von Vorteil ist vielmehr das Fehlen binärer Abhängigkeiten zwischen den eigentlichen Paket*definitionen*.

Als *Paket* bezeichnen die verschiedenen Linux-Distributionen ein Stück installierbare Software. Die Verwaltung der Pakete – vor allem also die Installation bzw. das Entfernen der Software – übernimmt der *Paketmanager*. Am

weitesten verbreitet ist RPM (*Red Hat Package Manager*), der RPM-Dateien als Pakete verwendet.

Bei Gentoo heißt der Paketmanager *Portage*, aber es werden keine zu RPM vergleichbaren Paketdateien genutzt. Hier liegt auch der grundsätzliche Unterschied zwischen einer auf vorkompilierten Programmpaketen basierenden Distribution wie Debian oder SUSE und einer quellbasierten Distribution wie Gentoo. Grundlage eines Paketes ist bei beiden Distributionstypen die Paketdefinition, die alle notwendigen Informationen für die Installation einer Software enthält. Dazu gehören z. B. der Link zu den Quellen, spezifische Anweisungen für das Entpacken und das Kompilieren des Paketes sowie Informationen zu Abhängigkeiten von anderen Paketen.

Ein binäres RPM-Paket wird vom Hersteller einer Distribution erstellt, indem die Software auf Basis dieser Paketdefinition vorkompiliert und als installierbares Paket zusammengepackt wird. Dem Nutzer kann das fertige Paket dann zum Download und der Installation in der eigenen Distribution zur Verfügung gestellt werden. Beim Kompilieren der Software entstehen zwangsläufig zusätzliche Abhängigkeiten, die der Paketmanager auf der Seite des Nutzers zwingend einhalten muss.

Im Gegensatz dazu besteht ein Gentoo-Paket eigentlich nur aus der Paketdefinition. Das Kompilieren der Software wird auf die Seite des Benutzers verlagert, und die Distribution liefert nur die Definitionen auf Source-Code-Ebene. Die binären Abhängigkeiten entstehen bei der Installation auf dem System des Nutzers und sind damit zwangsläufig erfüllt. Der Paketmanager Portage braucht sich nicht bzw. kaum um sie zu kümmern. Dies sorgt im Vergleich zu den bekannten Distributionen, bei denen vorkompilierte Pakete ausgeliefert werden, für eine deutlich höhere Flexibilität.

So folgt ein Gentoo-System dem Prinzip: „Install once, never reinstall.“ Die Aktualisierung des Systems findet kontinuierlich statt, und es gibt keinen Zustand, an denen Updates eine vollständige Neuinstallation des Systems notwendig machen. Man darf zwar nicht vergessen, dass die Aktualisierungen bei Gentoo in kleinen Häppchen über einen längeren Zeitraum verteilt werden und somit der Zeitbedarf auch nicht zwingend geringer ist als der für die gelegentliche Neuinstallation. Aber der Prozess ist deutlich flexibler und vielfach besser handhabbar, was vor allem für den Betrieb von Produktivsystemen ein wichtiger Vorteil sein kann. Möglich wird dieses Vorgehen dadurch, dass selbst die zentralen Bestandteile eines Linux-Systems (`glibc`, `gcc` etc.) nicht fest voneinander abhängen und so nicht in regelmäßigen Abständen gemeinsam aktualisiert werden müssen.

Das bedingt generell eine bessere Erweiterbarkeit des Systems. Die Kompatibilität verschiedener Softwareversionen ist deutlich größer, wenn aus dem Source-Code kompiliert wird, so dass sich ein stabiles Grundsystem aus etwas älteren Softwareversionen durchaus mit den neuesten Paketen in einem ausgewählten Bereich verträgt. Man muss sich nicht – wie z. B. bei

Debian – für ein stabiles oder ein eher experimentelles Grundsystem entscheiden; möchte man einzelne Software in der allerneuesten Version nutzen, muss man sich nicht mit generellen Problemen eines insgesamt wenig getesteten Systems beschäftigen, sondern verlässt sich auf eine stabile Basis und wählt instabile Pakete nur in Bereichen, in denen man hoffentlich selbst das nötige Wissen besitzt, um mit Problemen umzugehen.

Die Paketdefinitionen auf Source-Code-Basis ermöglichen es außerdem, eigene Modifikationen einzubringen. Damit bietet sich das System vor allem für Entwickler an, die bei Bedarf in alle Teile des Systems eingreifen. Da dies bei Gentoo geht, ohne das Paketmanagementsystem zu umgehen, bleiben solche Änderungen nicht reine Hacks, die beim nächsten Update verloren gehen; stattdessen lassen sie sich in die Paketdefinition einarbeiten und stehen auch längerfristig zur Verfügung. Das funktioniert selbst bei zentralen Bibliotheken recht gut.

Darüber hinaus ist die sehr gute Dokumentation zu erwähnen: Einerseits die Einführung des Gentoo-Dokumentationsprojekts auf der Projekt-Website[1], die die Installation und die grundlegende Benutzung abdeckt. Zudem findet man im englischen[2] und im deutschen[3] Gentoo-Wiki eine gute Auswahl aktueller Artikel zu verschiedenen Aspekten des Gentoo-Systems. Die dort gesammelten Informationen sind an vielen Stellen nicht mehr Gentoo-spezifisch, sondern als allgemeine Linux-Referenz brauchbar. Ebenfalls erwähnt seien die gut frequentierten Gentoo-Foren,[4] in denen Anfänger wie Profis Antworten auf ihre Fragen finden.

Nicht zuletzt stellt Gentoo ein ideales Lernsystem für Linux dar. Da jedes Paket aus dem Source-Code erstellt wird, hat der Nutzer die Möglichkeit, in wirklich jeden Teilbereich des Linux-Systems hineinzuschauen und den Ablauf zu verstehen. Bei binären Distributionen geht der Distributor den Schritt vom Source-Code zum binären Paket außerhalb des normalen Paketmanagements. Nutzer solcher Distributionen können ihn daher nur mit erhöhtem Aufwand nachvollziehen, während dieser Schritt bei Gentoo integraler Bestandteil der Paketinstallation ist.

Vor allem der letztgenannte Aspekt hat mich persönlich zu Gentoo geführt. Nachdem mir der erste Schritt von Windows zu SuSE geglückt und stabiles und konstantes Arbeiten unter Linux möglich war, ohne für bestimmte Anwendungen das Betriebssystem zu wechseln, überraschte mich die Erkenntnis, dass sich meine Erwartung, bei Linux alle Elemente des Systems verstehen zu können, nicht ganz erfüllte. Es war für mich als Linux-Anfänger nicht einfach zu verstehen, welche Grundkomponenten das System eigentlich ausmachen.

[1] http://www.gentoo.org/doc/
[2] http://gentoo-wiki.com/Main_Page
[3] http://de.gentoo-wiki.com/Hauptseite
[4] http://forums.gentoo.org/

In dieser Situation fand ich das Projekt *Linux from Scratch*[5] sehr hilfreich. Dessen Ziel ist eine Anleitung, ein System von Null auf Basis des Quellcodes zu erstellen. Hier bekam ich das erste Mal eine Ahnung davon, wie sich ein Linux-System eigentlich zusammensetzt.

Das löste zwar eine gewisse Begeisterung aus und hat mein Verständnis für Linux nachhaltig vorangebracht, aber das Vorhaben, meine Rechner mit Linux from Scratch zu installieren und zu pflegen, erwies sich schnell als Ding der Unmöglichkeit. Versucht man einmal, Linux ohne Paketmanagement zu verwenden, wird einem schnell klar, warum diese Art von Tools essentiell für den nervenschonenden Betrieb eines Linux-Systems sind.

Das war der Moment, in dem ich Gentoo für mich entdeckt habe und plötzlich vollauf zufrieden war, da ich in jeden Teilbereich des Systems schauen und auch eingreifen konnte. Gleichzeitig ist das System komfortabel zu verwalten und das Software-Angebot hochaktuell.

Der ersten Gentoo-Maschine folgten schnell weitere, und aus diesen anfänglichen Schritten entstand eine feste Bindung an diese Linux-Variante, die mich letztlich dazu führte, mich als aktiver Entwickler an der Erstellung der Distribution zu beteiligen.

Die Nachteile

Natürlich bringt Gentoo auch handfeste Nachteile mit sich: Der hohe Zeitaufwand für das Kompilieren der Software wurde bereits genannt. Gerade bei Desktop-Maschinen passt es selten, dass das Update einen halben Tag lang den Großteil der Rechnerressourcen belegt. Während sich die Aktualisierung kleinerer Pakete aufteilen lässt, geht das vor allem bei Desktop-Applikationen wie dem X-Server, KDE, OpenOffice oder Firefox, die wahre Zeitfresser sind, nicht. OpenOffice und Firefox lassen sich zwar auch als binäre Pakete installieren, für den X-Server und KDE gilt das aber nicht.

Als weiteres Problem erweisen sich häufig der natürliche Spieltrieb und der Wunsch nach den neuesten Softwarekomponenten. Sicher viele haben im ersten Übermut begonnen, ihr System mit dem Keyword `~x86` zu installieren (siehe Kapitel 5.3 ab Seite 130). Damit bekommt man zwar ein technisch hochaktuelles System, aber spätestens beim zwanzigsten Bug, der dazu führt, dass man sich mit den Innereien einer obskuren Bibliothek beschäftigen muss, lässt die Freude am System deutlich nach. Wer mit Gentoo produktiv arbeiten möchte ist gezwungen, seinen Spieltrieb im Zaum zu halten und als Basis die stabilen Pakete zu wählen.

Die instabilen Varianten sind nicht ohne Grund als instabil markiert: Während das Paket für sich genommen meist sogar funktioniert, führt die Interaktion mit anderen installierten Paketen zu Kollisionen. Selbst mit großem

[5] `http://www.linuxfromscratch.org/`

Linux-Know-how macht es wenig Spaß – und ist auch nicht besonders sinnvoll –, sich mit solch abseitigen Problemen herumzuärgern.

Darum sei hier einleitend vermerkt, dass man mit Gentoo unter keinen Umständen ein instabiles System betreiben sollte, denn damit beraubt man sich letztlich auch des Gentoo-spezifischen Vorteils, stabile und instabile Pakete miteinander kombinieren zu können und Probleme auf Bereiche zu beschränken, die man auch beherrscht.

Gentoo für wen?

Nicht jeder Nutzer wird Gentoo in der Handhabung lieben. Wie bei vielen Software-Systemen hängt dies im Wesentlichen vom eigenen, bevorzugten Arbeitsmodus ab.

Wer gerne in die Innereien eines Systems schaut und schon etwas Erfahrung mit Linux besitzt, wird sich vermutlich schnell heimisch fühlen.

Entwickler, die an verschiedenen Komponenten arbeiten, ohne das Paketmanagementsystem verlassen zu wollen, und die ihre Arbeit gerne auch Nutzern komfortabel zur Verfügung stellen, dürften Gentoo rasch schätzen lernen.

Schließlich dürften viele Linux-Nutzer in Gentoo eine interessante Alternative zu den Paketmanagementsystemen anderer Distributionen finden.

Und zuletzt sollte nicht unerwähnt bleiben, dass auch handfeste wirtschaftliche Gründe für Gentoo sprechen können – schließlich laufen alle Server meiner Firma unter eben diesem einen Betriebssystem.

Aus welchen Gründen auch immer Sie sich mit Gentoo beschäftigen, ich hoffe, dieses Buch ist Ihnen ein guter Begleiter!

Dank

Bedanken möchte ich mich bei den Menschen, die mich zur Arbeit an diesem Buch ermutigt und die auf ganz unterschiedliche Weise Anteil an seinem Erscheinen haben.

Dazu gehören allen voran meine Lektoren Dr. Markus Wirtz und Patricia Jung, von denen nicht nur die Idee zu diesem Buch ausging. Ich habe es zugegebenermaßen genossen, mein Fachchinesisch in lesbaren Text umformuliert zu bekommen. Bedanken möchte ich zudem für die Geduld, die sie mir bei der Bearbeitung des Manuskriptes entgegengebracht haben.

Auf Gentoo-Seite möchte ich mich bei Renat Lumpau und Stuart Herbert bedanken. Beide haben vor Jahren meine ersten Schritte als Gentoo-Entwickler begleitet und mir mit Rat und Tat zur Seite gestanden.

Darüber hinaus gilt mein Dank aber auch jedem einzelnen, der an Gentoo beteiligt ist – seien es nun die Entwickler oder die Nutzer, so dass der Dank im Grunde die gesamte „freie-Software-Gemeinde“ einschließt. Denn dass für mich ein Traum in Erfüllung gegangen ist und ich mit dem Programmieren freier Software meinen Lebensunterhalt verdienen kann erfüllt mich gelegentlich immer noch mit Staunen. Und das verdanke ich jedem, der – in welcher Weise auch immer – zu freier Software beiträgt.

Durch das Projekt begleitet und liebevoll unterstützt hat mich außerdem das Nordlicht, das hier oben in Hamburg auf seine ganz eigene Weise erstrahlt.

Gunnar Wrobel Hamburg, Januar 2008

Kapitel

1 Installation

Ohne grafischen Installer hat eine Distribution heute schnell den Ruf der Antiquiertheit oder mangelnder Benutzerfreundlichkeit. Entsprechend bietet auch Gentoo seit einiger Zeit ein GUI-basiertes Installationsprogramm, das auf der beiliegenden LiveDVD auch die Standardinstallationsmethode ist. Zwei gewichtige Gründe sprechen allerdings gegen dessen Einsatz:

- Die Installer-Software ist noch nicht ausreichend stabil, um grundsätzlich eine reibungslose Installation zu gewährleisten.
- Der Installer kaschiert das Handling der Gentoo-Basiskonfiguration, mit dem wir uns in diesem Buch explizit auseinandersetzen wollen.

Insbesondere aus dem zweiten Grund widmen wir uns in diesem Kapitel der „manuellen“ Installation. Dabei geht es weniger darum, alle Eventualitäten bei den ersten Schritten mit Gentoo abzudecken als den Installationsablauf dazu zu nutzen, einige grundlegende Konzepte der Distribution zu vermitteln und das System erst einmal ans Laufen zu bekommen. Den grafischen Installationsprozess beschreiben wir dann kurz im Anhang (ab

Seite 375); er sollte jedoch nicht ohne das in diesem Kapitel vermittelte Grundwissen gewählt werden.

Gentoo lässt sich auf einer Vielzahl Architekturen auf verschiedene Arten und Weisen installieren. Auch die Aufgaben des neuen Systems können die Vorgehensweise bei der Installation beeinflussen. Statt des zum Scheitern verurteilten Versuchs, alle denkbaren Varianten abzudecken, beschreiben wir im Folgenden, wie man konkret einen Webserver auf einem üblichen i686-System aufsetzt.

Die eigentlichen Installationsanweisungen finden sich auch im Internet[1] und treten in der folgenden Beschreibung etwas in den Hintergrund. Hier geht es darüber hinaus um Hintergrundinformationen, die über einen langen Zeitraum gültig und hilfreich sind.

1.1 Das System per DVD zum Leben erwecken

Die beigelegte DVD enthält die Gentoo 2007.0 LiveDVD für i686-Systeme, die ein bootfähiges System und alles Notwendige enthält, um Gentoo zu installieren. Sie lässt sich auf den meisten Rechnern problemlos starten. Als Mindestanforderung benötigt diese LiveDVD allerdings 256 MB Hauptspeicher. Wer auf einem sehr kleinen System installieren möchte, sollte die Variante `Minimal Install` von einem der Gentoo-Mirror-Server herunterladen[2] und auf CD brennen. Auf den Mirror-Servern finden sich auch die Installationsmedien für andere Rechnerarchitekturen, wie z. B. die PowerPC-Prozessoren, die Sparc-Maschinen von Sun oder auch die Alpha-Plattform. Verwendet man ein anderes Installationsmedium als die hier beschriebene LiveDVD sollte man allerdings das entsprechende Gentoo-Handbuch zur Hand haben, da vor allem auf anderen Prozessorarchitekturen einige Handgriffe anders aussehen.

Hinweis für 64-Bit-Systeme

Auch für eine 64-Bit-Installation muss man sich von den Mirror-Servern das entsprechende Image besorgen. Wir gehen hier zwar von einer (noch) üblichen 32-Bit-Installation aus, werden aber an geeigneten Stellen darauf hinweisen, was bei 64-Bit-Installationen gegenüber x86-Systemen zu beachten ist.

Kurz nach dem Start des Rechners mit der im Laufwerk befindlichen DVD sollte die Eingabeaufforderung erscheinen:

[1] `http://www.gentoo.org/doc/de/handbook/handbook-x86.xml?full=1`
[2] `http://www.gentoo.org/main/en/mirrors.xml`

```
boot:
```

Wir könnten nun mit der Return-Taste bestätigen und würden automatisch in die grafische Benutzeroberfläche des X-Servers gelangen. Wir gehen aber, wie schon erwähnt, einen anderen Weg und wählen den normalen Kernel (`gentoo`), aber mit der Option `nox` („keinen X-Server starten“), also:

```
boot: gentoo nox
```

Sollte der Boot-Prozess nach Auswahl des Standardkernels im Startmenü aus unerklärlichen Gründen stoppen, kann man sich in der Hilfe über weitere verfügbare Start-Optionen informieren. Das Deaktivieren einzelner Prozesse zur Hardware-Erkennung kann bei Boot-Problemen nützlich sein. Die entsprechenden Informationen erreicht man über die Tasten `[F1]` bis `[F7]`.

Probleme in diesem Stadium resultieren in den allermeisten Fällen daraus, dass der Kernel mit der Hardware der Maschine nicht zurecht kommt. Vielfach helfen die richtigen Kernel-Optionen an dieser Stelle weiter und schon ein einfaches

```
acpi=off
```

kann über das Ausschalten der ACPI-Unterstützung (*Advanced Configuration and Power Interface*) den Boot-Prozess plötzlich möglich machen.

Ernsthafte Probleme sind durch die mittlerweile sehr breite Hardware-Unterstützung des Linux-Kernels eher selten geworden, können aber, falls sie auftreten, ganz unterschiedliche Symptome zeigen, so dass wir sie hier nicht diskutieren können. Die schnellste Hilfe bietet in solchen Fällen das Gentoo-Forum.[3] Dort hat mit sehr hoher Wahrscheinlichkeit ein anderer schon einmal mit dem gleichen Mainboard, dem gleichen RAID-Controller oder der gleichen Grafikkarte dieselben Probleme gehabt und nennt die notwendigen Kernel-Optionen, die das System problemlos starten lassen.

Verwendet man eine Maschine mit sehr neuer Hardware und gibt es schon ein aktuelleres Gentoo-Release als die hier behandelte Version 2007.0, so sollte man dieses herunterladen und einsetzen.

Gibt es keine Probleme, erscheint nach kurzer Zeit die Auswahl der *Keymap*, über die man das verwendete Tastaturlayout einstellt. Für die deutsche Tastatur wählen Sie hier die `10`:

```
<< Load keymap (Enter for default): 10
```

Der Rest des Boot-Prozesses sollte automatisch ablaufen und Sie zuletzt auf die Kommandozeile der LiveDVD befördern:

[3] `http://forums.gentoo.org/`

```
livecd ~ #
```

Sollten Sie am Anfang des Boot-Vorgangs die Eingabe der `nox`-Option vergessen haben und in die grafische Benutzeroberfläche gelangt sein, können Sie jederzeit mit der Tastenkombination [Alt] + [F1] auf die Kommandozeile springen (und mit [Alt] + [F7] wieder zurück zum Desktop).

Haben Sie während des Boot-Vorgangs kein bzw. das falsche Tastaturlayout gewählt, können Sie es mit folgendem Befehl nachträglich auf das deutsche Layout setzen (siehe auch Seite 183):

```
livecd ~ # loadkeys de-latin1-nodeadkeys
Loading /usr/share/keymaps/i386/qwertz/de-latin1-nodeadkeys.map.gz
```

Wenn man beim Booten keine Angabe gemacht hat, wählt das System automatisch das US-amerikanische Tastaturlayout und für das oben angegebene Kommando liegt dann das y auf der Taste [z] und das Minus (-) findet sich als [ß] wieder.

1.2 Netzwerk oder kein Netzwerk?

Gentoo ist ein schnelllebiges System, das stark von einer Netzwerkanbindung profitiert. Prinzipiell kann man bei einer funktionierenden Verbindung in das Internet schon während der Installation aktuellere Pakete herunterladen und installieren.

Das stellt uns aber für dieses Buch vor ein Problem: Wenn Sie Ihr System zum frühest möglichen Zeitpunkt auf den aktuellsten Stand bringen, werden sich Unterschiede zwischen den von uns angegebenen Schritten und der Situation auf dem System ergeben. In den meisten Fällen werden sich nur die Versionsnummern unterscheiden, aber in Einzelfällen können die notwendigen Kommandos abweichen. Damit lässt sich nicht mehr garantieren, dass Sie dieses Buch exakt nach unseren Angaben durcharbeiten können.

Gentoo lässt sich aber mit der beigelegten DVD komplett ohne Netzwerkverbindung installieren, so dass Sie das hier Beschriebene ortsunabhängig testen und nachvollziehen können. Besonders für Gentoo-Anfänger halten wir es für sehr wichtig, dass dabei die hier angegebenen Kommandos eins zu eins umgesetzt werden können. Wir gehen darum im Wesentlichen von einer netzwerklosen Installation aus, damit wir Sie problemlos durch das Buch geleiten können.

Aber wir kommen natürlich nicht umhin, dass der Platz auf einer DVD begrenzt ist. Gentoo unterstützt derzeit über zehntausend verschiedene Softwarepakete, die beim besten Willen nicht alle auf der DVD Platz finden. Je

nach Kontext werden wir also auch gelegentlich Pakete vorstellen, für deren Installation Sie die Verbindung ins Internet benötigen. Wir werden an den entsprechenden Stellen darauf hinweisen und haben darauf geachtet, dass Sie diese Abschnitte in der Anfangsphase erst einmal überspringen können.

Sobald Sie ein wenig mit dem System vertraut sind, es auf den neuesten Stand gebracht haben und sich für die zusätzlichen Pakete interessieren, können Sie diese später problemlos installieren und testen. Und keine Sorge: Auch wenn Sie am Anfang nicht gleich mit den allerneuesten Paketversionen starten, Sie werden am Ende über ein aktuelles System verfügen. Das ist bei Gentoo das geringste Problem.

Falls Sie sich schon gut mit Gentoo auskennen oder sich nicht davon abschrecken lassen wollen, dass mal ein Befehl leicht abweichen kann, können Sie natürlich auch von Anfang an eine Netzwerkverbindung nutzen und mit einem aktuellen System starten.

Wir hoffen, dass wir so allen Ansprüchen an dieses Buch gerecht werden, und kommen zum ersten Hinweis in Punkto Netzwerkverbindung:

Hinweis für Systeme mit Netzwerkanbindung

Im optimalen Fall erkennt der Kernel die Netzwerkverbindung beim Booten automatisch. Ob schon eine Netzwerkverbindung besteht, sehen Sie über den Befehl `ifconfig`, der die Netzwerkkonfiguration anzeigt:

```
livecd ~ # ifconfig
eth0    Link encap:Ethernet  HWaddr 00:19:DB:A4:59:B9
        inet addr:192.168.178.21  Bcast:192.168.178.255
        Mask:255.255.255.0
        inet6 addr: fe80::219:dbff:fea4:59b9/64 Scope:Link
        UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500
        Metric:1
        RX packets:78 errors:0 dropped:0 overruns:0 frame:0
        TX packets:72 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:10988 (10.7 Kb)  TX bytes:11771 (11.4 Kb)
        Interrupt:21 Base address:0xa000
...
```

Dem Netzwerkinterface `eth0` ist hier die Netzwerkadresse `192.168.178.21` zugewiesen. Neben der Schnittstelle `eth0` sollte `ifconfig` noch `lo` anzeigen – diese existiert auf jedem Linux-Rechner auch ohne funktionierendes Netzwerk und stellt eine Verbindung des Systems zu sich selbst dar.

Hat der Kernel keine Netzwerkkarte automatisch erkannt, obwohl der Rechner Zugriff auf ein Netzwerk haben sollte, sei auf das Kapitel 3 ab Seite 77 verwiesen, wenn man den Server unbedingt schon bei der Installation mit dem Netzwerk verbinden möchte.

Auf der DVD finden sich unter `/mnt/cdrom/Getting_Online.txt` ebenfalls (allerdings englische) Instruktionen, um den Rechner mit dem Netzwerk zu verbinden. Diese lassen sich jederzeit mit `less` lesen:

```
livecd ~ # less /mnt/cdrom/Getting_Online.txt
```

Noch ein Tipp: Sollte das Netzwerk hier verfügbar sein, kann man an dieser Stelle auch den SSH-Server starten und die komplette Installation von einem anderen Rechner aus durchführen. Das ist die Variante, die ich persönlich bevorzuge. Dafür müssen wir nur den Dienst starten und über `passwd` ein Root-Passwort vergeben:

```
livecd ~ # /etc/init.d/sshd start
 * Generating Hostkey...
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
f5:3e:c9:e0:88:df:fd:3c:6f:3d:d1:17:d9:95:e3:dd root@livecd
 * Generating DSA-Hostkey...
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
02:9b:1f:2c:20:4c:9d:6e:da:f5:49:a1:4f:0e:f9:91 root@livecd
 * Generating RSA-Hostkey...
Generating public/private rsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_rsa_key.
Your public key has been saved in /etc/ssh/ssh_host_rsa_key.pub.
The key fingerprint is:
f5:8e:96:da:a9:fb:82:72:1f:7f:17:ad:ec:1f:07:53 root@livecd
 * Starting sshd ...
livecd ~ # passwd
New UNIX password: geheimes_rootpasswort
Retype new UNIX password: geheimes_rootpasswort
passwd: password updated successfully
```

Wie wir einen Dienst mit den Skripten in `/etc/init.d` starten können, erklärt Kapitel 7.3 ab Seite 170 ausführlicher. An dieser Stelle geht es uns zunächst einmal um die Möglichkeit, uns von einem entfernten Rechner aus zu verbinden, denn über SSH können wir uns nun von jedem anderen System aus mit dem festgelegten Passwort einloggen und die nachfolgenden Schritte von dort aus durchführen.

Für die Verbindung verwenden wir die IP-Adresse, die `ifconfig` oben (siehe Seite 25) für den zu installierenden Rechner ausgespuckt hat:

```
livecd ~ # ssh root@192.168.178.21
The authenticity of host '192.168.178.21 (192.168.178.21)' can't be
established.
RSA key fingerprint is f5:8e:96:da:a9:fb:82:72:1f:7f:17:ad:ec:1f:07:53.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.178.21' (RSA) to the list of known
```

```
hosts.
Password: geheimes_rootpasswort
```

`ssh` warnt uns davor, dass wir den Rechner derzeit nicht kennen, was aber unproblematisch ist, da wir die Maschine ja gerade erst einrichten. Als Passwort geben wir das Kennwort an, das wir kurz vorher auf der neuen Maschine angegeben haben.

1.3 Partitionen und Dateisysteme erstellen

In einem ersten Schritt bereitet man nun die Festplatte mit dem auch von anderen Distributionen verwendeten Standard-Tool `fdisk` für die eigentliche Installation vor. Wenn Sie noch nie mit `fdisk` gearbeitet haben, beschreibt das Gentoo-Installationshandbuch das Tool und die Partitionierung sehr ausführlich.

Hinweis für Systeme mit Netzwerkanbindung

Die aktuelle Version findet sich im Internet[4] und kann abgerufen werden, wenn das frisch gebootete System über eine Verbindung zum Internet verfügt:

```
livecd ~ # links http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?
part=1&chap=4#fdisk
```

Auch eine deutsche Version[5] ist verfügbar, wobei die englische Variante in der Regel etwas aktueller ist.

Die Möglichkeiten der Partitionierung sind unbegrenzt, und so kann diese mehr oder minder kompliziert ausfallen. Wir gehen hier von der einfachsten denkbaren Lösung aus: Gentoo soll auf einer Festplatte aufgespielt werden, die als erster IDE-Master angeschlossen ist. Diese Festplatte werden wir zudem möglichst einfach partitionieren. Weiter unten (ab Seite 30) finden sich dann Hinweise für komplexere Partitionierungsvarianten. Im Anhang B ab Seite 381 beschreiben wir, wie sich Gentoo auf einer Windows-Maschine unterbringen lässt, ohne das schon bestehende System zu beeinträchtigen.

[4] http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=1&chap=4#fdisk

[5] http://www.gentoo.org/doc/de/handbook/handbook-x86.xml?part=1&chap=4#fdisk

Die als erster IDE-Master angeschlossene Platte wird entweder über `/dev/hda` oder `/dev/sda` identifiziert. Wir gehen von `/dev/hda` aus und rufen `fdisk` entsprechend auf:

```
livecd ~ # fdisk /dev/hda
...
Command (m for help):
```

Auf unserem Referenzsystem legen wir mit dem `fdisk`-Befehl `n` („new“) eine kleine (20–50 MB) Boot-Partition vom Typ 83 (Linux), eine Swap-Partition (Typ 82) von 500–1000 MB und eine zweite, die restliche Festplatte umfassende Partition vom Typ 83 an.

Gehen wir zumindest kurz durch die notwendigen Tastaturkommandos, um die Festplatte nach dem oben angegebenen Schema zu partitionieren. Hat man `fdisk` gestartet, sollte man sich zuerst einmal alle vorhandenen Partitionen mit dem Befehl bzw. der Taste `p` anzeigen lassen. Diese kann man nun jeweils über die Tastaturfolge `d` („delete“) und die Nummer der Partition löschen, bis keine Partitionen mehr auf der Platte definiert sind und die Anzeige mit `p` nichts zurück liefert. `fdisk` löscht die Partitionen an dieser Stelle noch nicht wirklich, und es entsteht kein Datenverlust, wenn wir das Programm später mit `q` („quit“) beenden. Erst indem wir die neue oder veränderte Partitionierung mit `w` („write“) ausdrücklich auf der Festplatte verewigen, sind die Angaben wirksam.

Die Boot-Partition erstellen wir nun mit `n`, gefolgt von dem Partitionstyp „primär“, den wir mit der Taste `p` wählen. Mit der `1` definieren wir den Festplattenabschnitt als erste Partition. Als Startpunkt wählen wir den ersten Zylinder aus, indem wir einfach mit der Return-Taste bestätigen. Die Größe geben wir jetzt am besten in Megabyte mit einem vorangestellten Pluszeichen an: `+50M`.

Nun folgt die Swap-Partition, die wir in der gleichen Weise mit 1000 MB ausstatten und der wir die Nummer 2 geben (`n`, `p`, 2, `[Ret]`, `+1000M`). Um die Partition für das Swapping zu markieren, müssen wir den Typ 82 (Swap-Space) setzen, und zwar über das Kürzel `t` („type“), gefolgt von der Partitionsnummer 2 und dem Partitionstyp 82. `fdisk` sollte jetzt melden, dass der Partitionstyp auf `Linux Swap` gesetzt wurde.

Der letzten Partition spendieren wir den Rest der Platte und weisen die Partitionsnummer 3 zu (`n`, `p`, 3, zweimal `[Ret]`).

Damit ergibt sich folgende Tabelle als Grundlage unseres Beispielsystems:

```
Command (m for help): p
...
   Device Boot    Start        Ende      Blocks   Id  System
/dev/hda1             1           7       56196+  83  Linux
/dev/hda2             8         130      987997+  82  Linux Swap/Solaris
/dev/hda3           131        1869    13968517   83  Linux
```

Diese neue Partitionstabelle schreiben wir mit dem Befehl `w` endgültig auf die Platte. `fdisk` beendet sich anschließend auch automatisch und befördert uns auf die Kommandozeile zurück. Sollte man es sich anders überlegt haben, kann man `fdisk` jederzeit vorher über `q` beenden und damit alle Änderungen verwerfen. Nach dem `w` ist das nicht mehr möglich!

Um auf den beiden Linux-Partitionen nun ein Ext3-Dateisystem aufzubringen, nutzt man das Kommandozeilentool `mkfs.ext3`:

```
livecd ~ # mkfs.ext3 -L /boot /dev/hda1
mke2fs 1.38 (30-Jun-2005)
Filesystem label=/boot
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
14056 inodes, 56196 blocks
2809 blocks (5.00%) reserved for the super user
First data block=1
7 block groups
8192 blocks per group, 8192 fragments per group
2008 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 30 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.

livecd ~ # mkfs.ext3 -L / /dev/hda3
...
```

Die Option `-L` vergibt der Partition hierbei nur einen Namen (*Label*) und assoziiert sie noch nicht mit den hier angegebenen Verzeichnissen `/boot` oder `/`. Diese Bezeichner helfen aber besonders, wenn wir mehr als die drei vorgeschlagenen Partitionen angelegen möchten, da sie von `fdisk` im Partitionsschema mit angezeigt werden.

Die LiveDVD bietet Werkzeuge, um die bekanntesten Dateisysteme auf den Partitionen zu initialisieren. Wer z. B. Xfs bevorzugt verwendet den Befehl `mkfs.xfs` statt `mkfs.ext3`. Neben Ext3 und Xfs stehen unter anderem Ext2 (`mkfs.ext2`), JFS (`mkfs.jfs`), ReiserFS (`mkfs.reiserfs`) und verschiedene Windows-Dateisysteme (`mkfs.ntfs`, `mkfs.msdos`) zur Auswahl.

Auch den Swap-Bereich erzeugt und aktiviert man mit Standard-Werkzeugen, die auch auf anderen Distributionen nach der Installation zur Verfügung stehen:

```
livecd ~ # mkswap /dev/hda2
Setting up swapspace version 1, size = 1011703 kB
no label, UUID=32f64d1d-237a-4f4c-8814-33dedf48333b
livecd ~ # swapon /dev/hda2
```

1.3.1 Gentoo-spezifische Verzeichnisse mit höherem Platzbedarf

Eine große Partition, die das gesamte Root-Verzeichnis mit Ausnahme des `/boot`-Verzeichnisses umfasst, wie wir sie hier nutzen, vermeidet unnötige Größenbeschränkungen bei den einzelnen Unterhierarchien und reicht für die meisten Anwendungsfälle aus. Entscheidet man sich für eine komplexere Partitionierung, gibt es allerdings einige Gentoo-spezifische Verzeichnisse, die derart wachsen können, dass man sie unbedingt in die Überlegungen zur Partitionierung einbeziehen sollte.

Das wäre zum einen die Hierarchie unter `/usr/portage`. Hier lagert die komplette Paketdatenbank des Gentoo-Systems, zudem einige Verzeichnisse, die für die Funktionsweise des Paketmanagers Portage unerlässlich sind. Das Verzeichnis hat im Normalfall eine Größe von ca. einem halben GB.

Hier nicht eingerechnet sind zwei Ordner unterhalb von `/usr/portage`, die mit zunehmendem Alter des Systems deutlich anwachsen können: Das Verzeichnis `/usr/portage/distfiles` enthält die Quellpakete aller installierten oder zu installierenden Software. Beim Einspielen eines neuen Pakets lädt Portage das entsprechende Quellpaket in dieses Verzeichnis herunter. So wächst dieser Ordner im Schnitt auf zwei bis drei GB. Wie sich hier das Wachstum im Zaum halten lässt, beschreiben wir in Kapitel 11.1.4 ab Seite 257.

Das zweite Verzeichnis, `/usr/portage/packages`, beherbergt binäre Pakete, die Portage bei der Installation erstellt hat. Dies ist im Normalfall nur für so genannte Build-Hosts der Fall. Auch Binärpakete, die wir von bereits installierten Paketen erstellen (Kapitel 11.3.3), finden sich hier wieder. Die Größe dieses Verzeichnisses ähnelt der von `/usr/portage/distfiles`.

Damit benötigt der gesamte `/usr/portage`-Baum um die 3 GB für normale Systeme und um die 5 GB für Build-Hosts.

Ein weiteres Verzeichnis, das aufgrund seiner Nutzung durch Portage stark wachsen kann, jedoch außerhalb der `/usr/portage`-Hierarchie liegt, ist das temporäre Verzeichnis `/var/tmp`. Vor allem bei Desktop-Rechnern, auf denen Anwendungen wie KDE oder OpenOffice kompiliert werden, wächst dieses Verzeichnis während der Installation auf gut und gern 3–4 GB.

1.4 Das Grundsystem aufspielen

Die erstellten Partitionen bindet man nun unterhalb von `/mnt/gentoo` ins Installationssystem ein, um darauf das neue Gentoo-System einzurichten:

```
livecd ~ # mount /dev/hda3 /mnt/gentoo
livecd ~ # mkdir /mnt/gentoo/boot
livecd ~ # mount /dev/hda1 /mnt/gentoo/boot
```

Es empfiehlt sich, das Datum und die Uhrzeit des Systems zu überprüfen und gegebenenfalls zu korrigieren. Dazu dient der Befehl `date`.

An dieser Stelle haben wir noch nicht die Zeitzone festgelegt, in der wir uns befinden. Das geschieht erst später (siehe Seite 46), und derzeit rechnet die Maschine noch auf der Basis der Weltzeit (UTC). Für Deutschland wie für den übrigen mitteleuropäischen Raum gilt im Winter die Zeit UTC+1, und entsprechend ziehen wir gleich eine Stunde ab, wenn wir das Datum setzen. Während der Sommerzeit herrscht in Deutschland UTC+2, und es werden hier zwei Stunden abgezogen.

Nehmen wir an, es sei der 3. November 2007, 13:40 Uhr. Dann lässt sich die Uhr über `date` *MMDDhhmmCCYY* mit *MM* als Monat (hier: 11), *DD* als Tag (03), *hh* als Stunde (12 als 13 - 1, womit wir auf UTC umrechnen), *mm* als Minuten (40) und *CCYY* als Jahr (2007) folgendermaßen setzen:

```
livecd ~ # date 110312402007
Sat Nov  3 12:40:00 UTC 2007
```

1.4.1 Die Stage

Nun spielen wir das Grundsystem über zwei spezielle Pakete ein: die so genannte *Stage* und den Portage-Baum. Die Stage enthält alles, was man für ein funktionierendes (Gentoo-)Linux-System braucht, also vor allem den Compiler und die üblichen Kommandozeilenwerkzeuge. Stages stellt das Gentoo-Projekt in regelmäßigen Abständen als neue Gentoo-Releases zur Verfügung. Die beigelegte DVD enthält ein Stage-Archiv des 2007.0-Release, das wir im Folgenden verwenden werden, um an dieser Stelle nicht auf eine funktionierende Verbindung zum Internet angewiesen zu sein.

Hinweis für Systeme mit Netzwerkanbindung

Sollten allerdings schon neuere Releases verfügbar sein, kann man sich bei einer bestehenden Netzwerkverbindung auch das neueste Paket von den Gentoo-Servern herunterladen. Grundsätzlich empfehlen wir, an dieser Stelle erst auf die alte Version zu setzen und erst sehr spät, in Kapitel 10 wirklich auf die neueste Gentoo-Version zu wechseln. So haben Sie den Vorteil, den Erläuterungen im Buch exakt folgen zu können.

Wenn Sie sich trotzdem für ein neueres Release entscheiden, dann gelangen Sie mit dem Programm `links` auf die Liste der Download-Server:

```
livecd ~ # links http://www.gentoo.org/main/en/mirrors.xml
```

Auf jedem Mirror-Server (Seite 22) findet sich ein `releases`-Unterverzeichnis, in dem, sortiert nach Prozessor-Architektur und Release-Datum, die Stages im Verzeichnis `stages` angeboten werden. Historisch bedingt finden sich dort im Normalfall drei verschiedene Typen mit entsprechender Nummerierung:

Stage 1
: ein absolutes Basissystem. Bei einer Installation auf Grundlage dieser Stage wird Portage alle Pakete des Systems während der Erstinstallation neu kompilieren.

Stage 2
: hier liegen einige der zentralen Pakete wie z. B. `gcc` oder `glibc` bereits vorkompiliert vor.

Stage 3
: enthält alle Pakete eines Gentoo-Basissystems kompiliert. Bei Installation eines Stage-3-Archivs gelangt man also deutlich schneller zu einem funktionierenden Gentoo-System.

Allerdings berücksichtigen die in der Stage 3 enthaltenen Binärpakete eventuelle Vorgaben des Benutzers in Bezug auf die Compiler-Optionen nicht. Bei einer Stage-1-Installation werden hingegen alle Pakete neu kompiliert und entsprechend auch die Benutzervorgaben berücksichtigt. Da liegt die Schlussfolgerung nahe, dass man für ein durchoptimiertes System von einem Stage-1-System starten sollte. Der Gewinn eines solchen Vorgehens ist jedoch so gering, dass Gentoo diese Installationsmethode nicht mehr unterstützt.

Dies liegt vor allem daran, dass ein Gentoo-System veraltete Pakete bei einem Update ohnehin neu kompiliert. Der größte Teil der Pakete aus der Stage wird so in kürzester Zeit durch selbst kompilierte Pakete ersetzt, und das auf Stage-3-Basis installierte System unterscheidet sich dann in keiner Weise von einer Stage-1-Installation. Unter Berücksichtigung des Zeitgewinns, den Stage 3 bietet, und der Tatsache, dass aggressive Optimierung der Compiler-Optionen leicht zu Problemen führt, kommt man mit dem vorkompilierten Basissystem am besten weg.

Trotzdem bietet das `releases`-Verzeichnis der Mirror-Server alle drei Stages an. Das liegt vor allem daran, dass diese während des Release-Prozesses einer neuen Gentoo-Version ohnehin von den Entwicklern generiert werden. Außerdem gibt es Nutzer, die trotz des Zeitverlusts die Stage-1-Installation bevorzugen.

Wer sich hier eine Stage aus dem Internet herunterlädt, sollte diese auch, wie im nächsten Abschnitt auf Seite 34 beschrieben, mit `md5sum` und `gpg` auf ihre Integrität überprüfen.

Hinweis für 64-Bit-Systeme

Für die Installation eines 64-bit-Systems benötigen Sie eine Stage für die `amd64`-Architektur. Sie finden diese, wie oben angegeben, im `releases`-Verzeichnis unter `amd64`.

Doch zurück zur Stage der LiveDVD. Das dort verfügbare Archiv packt man einfach aus dem Archiv auf der DVD in die frisch formatierten Partitionen aus. Wer heruntergeladene, aktuellere Releases verwendet, muss den Dateinamen entsprechend anpassen.

```
livecd ~ # cd /mnt/gentoo
livecd gentoo # tar xjpvf /mnt/cdrom/stages/stage3-i686-2007.0.tar.bz2
...
livecd gentoo # cd ~
livecd ~ #
```

`tar` entpackt mit der Option `x` das Stage-Archiv. `j` wählt das Format des Archivs (`bzip2`), `p` erhält die Dateirechte während des Extrahierens und `v` zeigt uns, welche Dateien `tar` extrahiert. Zu guter Letzt gibt `f` an, dass wir die Datei mit dem nachfolgenden Dateinamen extrahieren möchten.

Mit `cd ~` bewegen wir uns wieder zurück in unser ursprüngliches Verzeichnis (`/root`).

1.4.2 Installieren der Gentoo-Paketdatenbank

Neben der Stage muss man außerdem die Gentoo-Paketdatenbank installieren, die wir hier auch von der DVD installieren werden. Die Paketdatenbank wird häufig auch als *Portage-Baum* bezeichnet.

Hinweis für Systeme mit Netzwerkanbindung

Die Paketdatenbank veraltet naturgemäß recht schnell, weshalb man durchaus auch zu einer aktuellen Version greifen kann. Wieder gilt allerdings die schon weiter oben angegebene Warnung: Sie entfernen sich damit an einigen Stellen von der folgenden Installationsanweisung.

Eine Möglichkeit besteht darin, die aktuellste Version von einem der Mirror-Server herunterzuladen. Sie befindet sich als nächtlicher Snapshot unter `snapshots`, und wir können sie, wie auch schon oben für die Stage angegeben, mit `links` herunterladen.

Verwendet man z. B. den Mirror der Ruhr-Universität Bochum, kann man sich auch mit folgenden `wget`-Befehlen den Snapshot und die notwendigen Prüfsummen herunterladen:

```
livecd ~ # wget http://linux.rz.ruhr-uni-bochum.de/download/gentoo-mirro
r/snapshots/portage-latest.tar.bz2
livecd ~ # wget http://linux.rz.ruhr-uni-bochum.de/download/gentoo-mirro
r/snapshots/portage-latest.tar.bz2.md5sum
livecd ~ # wget http://linux.rz.ruhr-uni-bochum.de/download/gentoo-mirro
r/snapshots/portage-latest.tar.bz2.gpgsig
```

Wir laden den neuesten Snapshot als Archiv `portage-latest.tar.bz2` mit der zugehörigen MD5-Prüfsumme (`portage-latest.tar.bz2.md5sum`) herunter und spielen ihn dann ins neue Gentoo-System ein, sofern seine Checksumme mit denen in den Prüfsummendateien übereinstimmt:

```
livecd ~ # md5sum -c portage-latest.tar.bz2.md5sum
portage-latest.tar.bz2: OK
```

Sollte der `md5sum`-Test fehlschlagen, so wurden Daten beim Herunterladen beschädigt und man lädt das Paket nochmals *von einem anderen Mirror-Server* herunter.

Damit haben wir die Integrität des Pakets getestet, aber noch nicht, ob das Paket wirklich vom Gentoo-Projekt stammt und ob es unverändert ist. Schließlich laden wir uns hier Daten herunter, bei denen es ein Leichtes wäre, sie so zu modifizieren, dass sie gleich unsere gesamte Festplatte löschen:

```
livecd ~ # gpg --recv-keys 7DDAD20D
gpg: keyring ‘/root/.gnupg/secring.gpg’ created
gpg: requesting key 7DDAD20D from hkp server subkeys.pgp.net
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 7DDAD20D: public key ‘‘Gentoo Portage Snapshot Signing Key
(Automated Signing Key)’’ imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:               imported: 1
livecd ~ # gpg --verify \
  portage-latest.tar.bz2.gpgsig portage-latest.tar.bz2
gpg: Signature made Tue Jul 24 01:58:09 2007 UTC using DSA key ID
7DDAD20D
gpg: Good signature from ‘‘Gentoo Portage Snapshot Signing Key
(Automated Signing Key)’’
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: 4AC0 D5FE 8F92 96BA 6A06  0A2A BB1D 301B 7DDA
D20D
```

Wichtig ist die Meldung `Good signature from "Gentoo Portage Snapshot Signing Key (Automated Signing Key)"`. `gpg` warnt uns zwar noch, dass wir in Bezug auf die Identität des `Gentoo Portage Snapshot Signing Key` nicht ganz sicher sein können, aber das ist an dieser Stelle vollkommen in Ordnung.

Alternativ zum Herunterladen eines Snapshots kann man den Portage-Baum auch über das ab Seite 40 beschriebene Verfahren aktualisieren. Die Menge an übertra-

genen Daten dürfte im Vergleich zu dem heruntergeladenen Archiv etwas geringer sein, aber das Auspacken des Archivs ist im Normalfall schneller erledigt.

Wir verwenden aber hier ebenfalls ein Archiv von der LiveDVD, um weiterhin ohne Netzwerkverbindung zu arbeiten. Wer das neueste Archiv heruntergeladen hat, modifiziert `portage-2007.0.tar.bz2` zu `portage-latest.tar.bz2`.

```
livecd ~ # tar xvjf /mnt/cdrom/snapshots/portage-2007.0.tar.bz2 -C \
/mnt/gentoo/usr
...
```

Diesmal entpacken wir das Archiv in das Verzeichnis `/mnt/gentoo/usr`, angegeben über die Option `-C`.

Mit der Paketdatenbank haben wir unserem System einen notwendigen Baustein für die Installation neuer Pakete hinzugefügt. Ergänzend zu den eigentlichen Paketdefinitionen benötigt Portage aber natürlich auch den Quellcode, aus dem das Paketmanagementsystem die Software installieren wird. Jede Paketdefinition enthält einen Link, unter dem das Quellarchiv eines Pakets heruntergeladen werden kann. Portage kann das bei bestehender Netzwerkverbindung automatisch erledigen. Wir wollen uns hier aber die Option offen halten, auch ohne Netzwerk arbeiten zu können, und müssen deshalb auch noch die Quellarchive von der DVD kopieren:

```
livecd ~ # cp -r /mnt/cdrom/distfiles /mnt/gentoo/usr/portage/
```

Die Option `-r` kopiert das Verzeichnis inklusive Inhalt.

1.5 Betreten der neuen Umgebung

Damit ist der Grundstein für unser neues System gelegt. Wir können es zwar noch nicht booten, aber es ist möglich, schon einmal aus dem System der LiveDVD in das neue System zu springen. Hierfür wechselt man über `chroot` in das Verzeichnis `/mnt/gentoo` und blendet damit aus dem derzeitigen System in die Neuinstallation über. Im Wesentlichen wechselt dabei das Root-Verzeichnis von `/` auf `/mnt/gentoo` – darum auch `chroot` („change root“).

Zuvor sind aber noch einige wenige Handgriffe notwendig, damit sich unsere neue Umgebung auch wirklich wie ein vollwertiges Linux-System verhält.

Hinweis für Systeme mit Netzwerkanbindung

Sollte sich unser System schon im Netz befinden, benötigen wir die Netzwerkinformationen auch im neuen System, um auch dort auf das Internet zugreifen zu können. Darum kopiert man die Datei `/etc/resolv.conf` aus dem Installationssystem in die Neuinstallation:

```
livecd ~ # cp /etc/resolv.conf /mnt/gentoo/etc/
```

Wir benötigen zwei spezielle, auf dem System der LiveDVD existierende Dateisysteme auch im neuen System: die Hierarchie der Gerätedateien und die Informationen aus dem `/proc`-Verzeichnis. Beide werden vom System bei einem normalen Boot-Vorgang automatisch eingebunden, aber da wir unser neues System ja noch nicht gebootet haben und nur mit `chroot` übergewechselt sind, fehlen diese noch.

```
livecd ~ # mount -o bind /dev /mnt/gentoo/dev
```

Über `-o bind` wird das Geräteverzeichnis `/dev` aus dem LiveDVD-System einfach in das neue System unter `/mnt/gentoo/dev` gespiegelt. Sobald wir mit `chroot` in die Umgebung `/mnt/gentoo` gewechselt haben, findet sich das Verzeichnis dann wie gewohnt als `/dev` in unserer Umgebung wieder.

Ähnliches gilt für das `/proc`-Verzeichnis, über das die Prozesse im System mit dem Kernel interagieren können. Da `/proc` ein spezielles Dateisystem darstellt, müssen wir den Typ mit `-t proc` angeben und gleichzeitig mit `none` auf die Angabe einer Quelle verzichten.

```
livecd ~ # mount -t proc none /mnt/gentoo/proc
```

So vorbereitet, können wir nun über `chroot` in die neue Umgebung wechseln:

```
livecd ~ # chroot /mnt/gentoo /bin/bash
livecd / #
```

Über `/bin/bash` geben wir hier an, dass wir in der neuen Umgebung auch die Bash als Kommandozeile verwenden wollen.

Übrigens können wir die neue Umgebung jederzeit mit `exit` verlassen und befinden uns anschließend wieder im System der LiveDVD.

Sollte es notwendig sein, die Arbeit zu unterbrechen, den Rechner abzuschalten und zu einem späteren Zeitpunkt erneut mit der LiveDVD zu starten, so kommt man jederzeit mit folgender Befehlsfolge wieder auf den alten Stand und kann fortsetzen, wo man stehen geblieben war:

```
livecd ~ # mount /dev/hda3 /mnt/gentoo
livecd ~ # mount /dev/hda1 /mnt/gentoo/boot
livecd ~ # swapon /dev/hda2
livecd ~ # mount -o bind /dev /mnt/gentoo/dev
livecd ~ # mount -t proc none /mnt/gentoo/proc
livecd ~ # chroot /mnt/gentoo /bin/bash
livecd / #
```

1.5.1 Umgebungsvariablen setzen

Wir befinden uns aber nun in einer Bash des neuen Systems. Diese definiert wie jede andere Shell im Normalfall auch einen bestimmten, mit dem Befehl `export` einsehbaren Satz an Umgebungsvariablen, die das Standardverhalten diverser Kommandozeilentools beeinflussen. In dem neuen System müssen wir diesen Satz aber zunächst einmal aktualisieren.

Neu installierte Pakete legen Variablendefinitionen, die die Bash für sie setzen soll, in Form kleiner Dateien in `/etc/env.d` ab. Genaueres zu diesem Verzeichnis findet sich in Abschnitt 9.1 ab Seite 192. Dieses Verzeichnis liest das Tool `env-update` aus. Es schreibt die Informationen aus dem Verzeichnis in die Datei `/etc/profile.env` um.

Deren Inhalt liest die systemweite Bash-Konfigurationsdatei `/etc/profile` per `source`-Befehl ein. Dieses Skript wird beim Login in eine neue Bash-Session auf einem Gentoo-System automatisch ausgelesen.

Im Normalfall ruft Portage `env-update` automatisch nach der Installation neuer Pakete auf, aber beim Wechsel in das frische System müssen wir die Umgebungskonfiguration einmal per Hand aktualisieren:

```
livecd / # env-update
>>> Regenerating /etc/ld.so.cache...
livecd / # source /etc/profile
```

Nun sind alle Vorbereitungen getroffen, um neue System entsprechend unseren Bedürfnissen zu konfigurieren.

1.6 Anpassungen der Portage-Konfiguration

Als eine der zentralen Konfigurationsdateien beeinflusst `/etc/make.conf` das Verhalten von Portage. Eine gut kommentierte Beispielkonfiguration findet sich in `/etc/make.conf.example`. Detailliertere Informationen zu `make.conf` finden sich auch im Kapitel 6 ab Seite 141. Es reicht für den Anfang aber, sich mit einer Handvoll der darin aufgeführten Variablen zu beschäftigen.

1.6.1 Compiler-Flags festlegen

Schauen wir uns einmal die Standardwerte in dieser Datei an:

```
livecd / # cat /etc/make.conf
# These settings were set by the catalyst build script that
# automatically built this stage.
# Please consult /etc/make.conf.example for a more detailed example.
CFLAGS="-O2 -mtune=i686 -pipe"
CXXFLAGS="${CFLAGS}"
# This should not be changed unless you know exactly what you are doing.
# You should probably be using a different stage, instead.
CHOST="i686-pc-linux-gnu"
```

Manche Werte können abhängig von der ausgewählten Stage variieren. Bei einem x86-Stage3-Archiv ist der CHOST-Wert z. B. auf i486-pc-linux-gnu gesetzt, was auf jeden modernen Prozessor passen sollte. Bei neueren Maschinen sollte man aber gleich das entsprechende i686-Archiv wählen, so dass, wie auch durch die Stage der LiveDVD vorgegeben, CHOST auf i686-pc-linux-gnu gesetzt ist.

Hinweis für 64-Bit-Systeme

Für eine 64-Bit-Installation auf x86-Systemen muss man ein amd64-Installationsmedium verwenden. Die CHOST-Variable muss dann auf x86_64-pc-linux-gnu gesetzt sein.

Den Wert für CHOST sollten wir auch nicht anders als von i486 auf i586 bzw. i686 verändern. Hat man den Eindruck, dass der Wert nicht stimmt, so verwendet man vermutlich das falsche Stage-Archiv.

Im einfachsten Fall belässt man auch die CFLAGS- und CXXFLAGS-Werte in der Datei beim Standard und verzichtet auf weitere Anpassungen. Aber gerade bei aktuelleren Prozessoren möchte man die neueren Eigenschaften der CPU natürlich auch nutzen, statt ein allgemein gültiges, aber langsameres System zu erhalten.

Wer die Datei bearbeiten möchte nutzt dazu nano als Editor. Er ist in diesem Stadium der einzig verfügbare Editor.

```
livecd / # nano /etc/make.conf
```

Die Navigation im Text erfolgt mit den Cursor-Tasten. Die Tasten-Kombination [Strg] + [O] speichert Modifikationen an der Datei, während

[Strg] + [X] nano beendet. Eine ausführlichere Beschreibung der verfügbaren Kommandos erhält man in der Hilfe, die man mit der Kombination [Strg] + [G] aufruft.

Die Variablen CHOST, CFLAGS und CXXFLAGS beeinflussen das Verhalten des Compilers gcc und legen in Form von Compiler-Flags fest, welcher Maschinentyp das übersetzte Endresultat verarbeiten kann. Die korrekten Einstellungen hängen entsprechend stark vom verwendeten Prozessor ab. Die Übersicht im Gentoo-Wiki[6] verrät, welche Variablenwerte für welche CPU passen.

Welche CPU man genau verwendet, verrät die Datei /proc/cpuinfo:

```
livecd / # cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 6
model           : 8
model name      : AMD Athlon(tm) XP 2000+
stepping        : 0
cpu MHz         : 1665.406
cache size      : 256 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 mmx fxsr sse syscall mmxext 3dnowext 3dnow up ts
bogomips        : 3334.60
```

vendor_id, cpu family, model, model name enthalten hier die ausschlaggebenden Werte und finden sich so auch auf der oben angegebenen Wiki-Seite wieder.

Auf x86-Systemen kann auch das Tool x86info hilfreich sein. Wir beschreiben es im Kapitel 16.5 ab Seite 353.

Im eigenen Interesse sollten nur sehr erfahrene Benutzer von den im Wiki vorgeschlagenen Einstellungen abweichen, denn der Einfluss der Compiler-Flags ist sehr komplex. Im einfachsten Fall führen inkompatible Einstellungen dazu, dass sich einzelne Pakete nicht kompilieren lassen und während der Installation abbrechen. In vielen Fällen wird allerdings „nur“ die Lauffähigkeit eines Programms in Mitleidenschaft gezogen: Man sieht sich mit Fehlern konfrontiert, die sich nicht sofort mit einer Fehlkonfiguration der gcc-Flags in Verbindung bringen lassen.

[6] http://gentoo-wiki.com/Safe_Cflags

Dass sich die Gentoo-Distribution nach dem schnellsten Schwimmer unter den Pinguinen nennt, begründet sich unter anderem mit der Möglichkeit, die Compiler-Flags so zu optimieren, dass der Rechner ein Maximum an Geschwindigkeit erreicht. Allerdings suggeriert dies auch, dass Optimierung zu den durchaus üblichen Vorgängen bei einer Gentoo-Installation gehört. Das ist nicht der Fall. Der Geschwindigkeitsgewinn beim Kompilieren mit aggressiver `gcc`-Optimierung liegt meist nur im unteren Prozent-Bereich und gleicht den Zeitverlust durch das Debuggen der verursachten Probleme nicht aus.

Wer zumindest bestimmte Pakete mit optimierten Flags kompilieren möchte, findet entsprechende Tipps in Kapitel 16.3.3 ab Seite 350. Dieser Ansatz destabilisiert nicht gleich das komplette System, sondern erlaubt es, bestimmte Software einzeln zu optimieren. Das lohnt sich z. B. für besonders oft und intensiv verwendete Komponenten eines Servers.

Nach der Basis-Konfiguration für Portage könnte man nun, vorausgesetzt es besteht eine Netzwerkverbindung, den Portage-Baum aktualisieren. Wir raten mit der schon oben genannten Begründung (siehe Seite 31) ab: Es würden nicht mehr alle Angaben mit dem Buch übereinstimmen, und es ist kein Problem, zu einem späteren Zeitpunkt auf den aktuellsten Stand zu wechseln. Dem Gentoo-Anfänger sei also geraten, das folgende Kapitel einstweilen zu überspringen.

Hinweis für Systeme mit Netzwerkanbindung

Um hier schon auf das aktuellste System zu wechseln, kann man die eigene Kopie des Portage-Baums an dieser Stelle mit der von den `rsync`-Servern zu beziehenden Version abgleichen.

Wer sich möglichst nah an den hier beschriebenen Anweisungen entlang arbeiten möchte, der verzichtet jedoch zunächst einmal auf die Aktualisierung und wechselt ab Kapitel 10 auf das neueste System.

Der folgende Aufruf bringt den Portage-Baum auf den neuesten Stand:

```
livecd / # emerge --sync
```

Er gleicht den lokal gespeicherten Baum per `rsync` mit der Version auf einem der regionalen Server ab. Wen der `rsync` geschuldete übermäßige Output stört, erweitert den `emerge`-Befehl um den Switch `--quiet`.

Sollte eine neue Portage-Version verfügbar sein, so meldet sich `emerge` nach Abschluss des Synchronisationsvorgangs mit der Nachricht

```
 * An update to portage is available. It is _highly_ recommended
 * that you update portage now, before any other packages are updated.
 * Please run 'emerge portage' and then update ALL of your
 * configuration files.
 * To update portage, run 'emerge portage'.
```

Als zentralen Baustein des Gentoo-Systems sollte man Portage tatsächlich vorzugsweise in der aktuellsten Version benutzen. Das Update auf die in dieser Meldung angekündigte neue Ausgabe führt man ebenfalls mit `emerge` durch. In der Tat besteht die Hauptfunktionalität des Befehls im Installieren, Updaten und Deinstallieren von Software-Paketen. Das `portage`-Paket selbst aktualisiert man mit dem kurzen Befehl:

```
livecd / # emerge sys-apps/portage
```

1.7 Was man über emerge wissen sollte

Kommen wir zum zentralen Werkzeug des Portage-Systems: `emerge`. Ohne weitere Optionen, schlicht unter Angabe des betreffenden Pakets aufgerufen, stellt `emerge` die einfachste Form der Gentoo-Paketinstallation und -aktualisierung dar. Was man als Argument angibt, ist allerdings eine etwas komplexere Angelegenheit.

1.7.1 Paketbezeichnung und -aufbau

Die korrekte Bezeichnung eines Gentoo-Pakets setzt sich aus der Kategorie, dem Paketnamen, der Version und der Revision zusammen. Die Kategorie sortiert die Pakete grob vor: Sie entspricht dem Namen eines Ordners unterhalb von `/usr/portage` (bzw. `PORTDIR`, siehe Kapitel 6.1 auf Seite 143). Hier gibt es zum Beispiel die Kategorie `app-portage` mit `portage`-spezifischen Anwendungen, während der Ordner `net-www` mit webbezogenen Applikationen oder `sys-apps` mit wichtiger Systemsoftware bestückt ist. Die Bedeutung der Kategorien erschließt sich vielfach aus der Kurzbezeichnung, aber jede Kategorie enthält auch eine Datei `metadata.xml`, die eine längere Beschreibung liefert. Hier z. B. ein Auszug aus `/usr/portage/app-portage/metadata.xml`:

```
livecd / # cat /usr/portage/app-portage/metadata.xml
...
<longdescription lang="de">
Die Kategorie app-portage enthält Programme für das Arbeiten mit Portage
oder Ebuilds.
</longdescription>
...
```

Innerhalb einer Kategorie repräsentieren einzelne Verzeichnisse die Einzelpakete. Der Name des Verzeichnisses entspricht dem Paketnamen. Dieser

muss innerhalb einer Kategorie eindeutig sein, nicht jedoch kategorienübergreifend. So beschwert sich der folgende `emerge`-Befehl darüber, dass der Paketname `muse` (erlaubterweise) sowohl in der Kategorie `app-emacs` als auch in `media-sound` vorkommt:

```
livecd / # emerge muse
Calculating dependencies

!!! The short ebuild name "muse" is ambiguous.  Please specify
!!! one of the following fully-qualified ebuild names instead:

    app-emacs/muse
    media-sound/muse
```

Während für die meisten Pakete die Verwendung des einfachen Namens ausreicht, ist der Nutzer in diesem Fall gezwungen, die Kategorie des gewünschten Pakets, durch / getrennt, ebenfalls anzugeben:

```
livecd / # emerge media-sound/muse
```

Schaut man sich den Inhalt eines Paketverzeichnisses an, so gibt es dort nicht nur eine Datei: Es enthält vielmehr die Definitionen für verschiedene Paketversionen, die so genannten *Ebuilds*:

```
livecd / # ls -la /usr/portage/sys-apps/portage/
total 80
drwxr-xr-x   3 root root  4096 Mar 12 14:41 .
drwxr-xr-x 216 root root  4096 Apr 12 21:35 ..
-rw-r--r--   1 root root 20867 Mar 12 14:41 ChangeLog
-rw-r--r--   1 root root  6755 Mar 12 14:41 Manifest
drwxr-xr-x   2 root root  4096 Mar 12 14:41 files
-rw-r--r--   1 root root   282 Mar 12 14:41 metadata.xml
-rw-r--r--   1 root root  5675 Mar 12 14:41 portage-2.0.51.22-r3.ebuild
-rw-r--r--   1 root root  7206 Mar 12 14:41 portage-2.1.1-r2.ebuild
-rw-r--r--   1 root root  6909 Mar 12 14:41 portage-2.1.2-r9.ebuild
-rw-r--r--   1 root root  6935 Mar 12 14:41 portage-2.1.2.2.ebuild
```

Deren Namen setzen sich nach dem Schema

paketname-version-revision.ebuild

zusammen. Der Paketname ist durch das Verzeichnis vorgegeben. Die Version entspricht im Normalfall der Versionsnummer der Software, so wie sie das entsprechende Software-Projekt zur Verfügung stellt. Ein Ebuild des Apache-Servers in der Version 2.0.54 heißt entsprechend `apache-2.0.54`.

Vielfach erfährt die Paketdefinition ebenfalls Verbesserungen, während das Software-Archiv selbst das gleiche bleibt. In diesen Fällen kennzeichnet man den Ebuild mit einer eigenen Revisionsnummer.

Der erste Ebuild eines Pakets trägt keine Revisionsnummer, darauf folgende Ebuild-Versionen, die das gleiche Ursprungsarchiv benutzen, bekommen die Markierung `-r1`, `-r2` etc. angehängt.

Der vollständige Name eines Pakets lautet also beispielsweise `sys-apps/portage-2.0.51-r3`. Dieses Paket lässt sich auf drei gleichwertige Weisen installieren:

```
livecd / # emerge portage
livecd / # emerge sys-apps/portage
livecd / # emerge =sys-apps/portage-2.0.51-r3
```

Die erste Variante scheitert, wenn es mehrere Pakete mit gleichem Namen gibt. Dies ist aber selten der Fall. Möchte man, wie im letzten Beispiel, eine konkrete Version installieren, muss man der Paketbezeichnung ein Präfix voranstellen (vgl. Kapitel 4.4 ab Seite 108). Das Gleichheitszeichen sagt in diesem Fall, dass wir genau die angegebene Version installieren wollen.

Wir werden in diesem Buch grundsätzlich die volle Paketbezeichnung (also `sys-apps/portage` statt `portage`) verwenden, um Zweideutigkeiten zu vermeiden und ein Gefühl dafür zu vermitteln, wie die verschiedenen Applikationen in die Kategorien des Portage-Baumes eingeordnet sind.

1.7.2 Generalprobe für die Installation

Zwei zentrale Switches verhindern bei Bedarf, dass `emerge` ein Paket sofort installiert, stattdessen zeigt `emerge` bei ihrer Verwendung nur an, was passieren *würde*:

```
livecd / # emerge -av sys-apps/portage

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] sys-apps/portage-2.1.2.2  USE="-build -doc -epydoc (-se
linux)" LINGUAS="-pl" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] No
```

Der Switch `-a` (auch: `--ask`) verhindert, dass die Software sofort installiert wird, während `-v` oder `--verbose` die Informationsdichte erhöht. Damit verrät `emerge`, welche Pakete Portage im Falle einer Installation noch einspielen würde, und vor allem, welche Features jedes dieser Pakete unterstützt. Nachdem `emerge` diese Informationen angezeigt hat, wartet das Programm nun, bis wir ihm mit `Yes` oder `No` mitgeteilt haben, ob wir die Pakete wirklich installieren möchten.

Auf diese beiden Optionen von `emerge` gehen wir auf Seite 97 auch nochmal genauer ein, werden sie im folgenden aber erst einmal für jede Paketinstallation verwenden.

1.8 Das Installationsprofil wählen

Die Grunddefinition eines zu installierenden Systems legt man bei Gentoo mit Hilfe so genannter *Profile* fest. Sie definieren z. B., welche Pakete für das Funktionieren der Installation unverzichtbar sind, welche Software unter keinen Umständen installiert sein darf, welches Paket den Zuschlag erhält, wenn mehr als eines eine gewünschte Funktionalität bereitstellt, und sie geben gewisse Eigenschaften vor, die neu eingespielte Pakete aufweisen sollen.

Bei der Wahl des eigenen Profils aus dem Angebot unter `/usr/portage/profiles` muss man die eigene Hardware-Architektur beachten.

So finden sich unter `/usr/portage/profiles/default-linux` die von Gentoo unterstützten CPU-Typen mit ihren spezifischen Kürzeln wieder: `alpha`, `amd64`, `arm`, `hppa`, `ia64`, `m68k`, `mips`, `ppc`, `ppc64`, `s390`, `sparc` und `x86`.

Für die überwiegende Zahl dieser Architekturen findet sich dann innerhalb des entsprechenden Verzeichnisses ein weiteres Verzeichnis mit dem Namen des aktuellsten Gentoo-Release, hier also `2007.0`. Für die besonders verbreitete `x86`-Architektur finden sich neben dem eigentlichen Release-Profil noch spezielle Varianten, wie z. B. das `xbox`-Profil, mit dem sich Gentoo auf der Xbox installieren lässt, oder auch das `vserver`-Profil, mit dem ein virtueller Server problemlos unter Gentoo läuft.

Wer ein Gentoo-System ohne Linux-Kernel wünscht, findet auf der obersten Ebene `/usr/portage/profiles` mit `default-bsd` ein Profil, das auf BSD-Basis arbeitet. Gentoo unterstützt derzeit allerdings nur die `x86`- und die `sparc`-Architektur.

Im Normalfall ist das Standard-Profil für die `x86`-Architektur voreingestellt, und zwar über einen symbolischen Link auf `/usr/portage/profiles/default-linux/x86/2007.0` im `/etc`-Verzeichnis:

```
livecd / # ls -g /etc/make.profile
lrwxrwxrwx 1 root 48 Jul  5 11:29 /etc/make.profile -> ../usr/portage/pr
ofiles/default-linux/x86/2007.0
```

Mit der Option `-g` für `ls` wird hier das Ziel der Verknüpfung angezeigt, nicht der Inhalt des Verzeichnisses, auf das verwiesen wird.

Fehlt die oben angegebene Verknüpfung oder wünscht man ein anderes Profil, korrigiert man den Verweis entsprechend:

```
livecd / # ln -snf /usr/portage/profiles/default-linux/x86/2007.0 \
/etc/make.profile
```

Beim Verknüpfen mit der Option `-s` erzeugt man einen symbolischen Verweis, und die Optionen `-n` und `-f` sorgen dafür, dass `ln` die Verknüpfung `/etc/make.profile` auch wirklich ersetzt und den Verweis nicht innerhalb des verwiesenen Verzeichnisses anlegt.

Hinweis für 64-Bit-Systeme

Das Profil für die 64-Bit-Installation lautet `default-linux/amd64/2007.0`, und entsprechend muss der obige Befehl folgendermaßen angepasst werden:

```
livecd / # ln -snf /usr/portage/profiles/default-linux/amd64/2007.0 \
/etc/make.profile
```

1.9 USE-Flags setzen

Die vorgegebenen Gentoo-Profile sind jedoch nicht so fein definiert, dass man sich hier z. B. bereits die Standardkonfiguration eines LAMP-Servers aussuchen könnte. Dafür setzen wir einige Einstellungen, so genannte *USE-Flags*, in der Datei `/etc/make.conf`, die dem Paketmanager vorgeben, dass er alle Programme, die bei entsprechender Vorkonfiguration durch `configure` für unsere Zwecke passende Funktionalität liefern, entsprechend kompilieren soll.[7]

Wie bereits erwähnt, stellt Gentoo in dieser frühen Phase der Installation nur `nano` als Editor bereit; entsprechend rufen wir `nano /etc/make.conf` auf, um nachfolgende Zeile hinzuzufügen:

```
USE="-X apache2 ldap mysql xml"
```

Da der Rechner als Webserver ohne X-Server und grafische Desktop-Anwendungen auskommt, deaktivieren wir mit dem Minus explizit das USE-Flag `X`. Das Setzen von `apache2` sorgt dagegen dafür, dass alle Pakete, die bei entsprechenden Einstellungen zur Compile-Zeit über Möglichkeiten verfügen, mit Apache 2 in irgendeiner Art und Weise zusammenzuarbeiten oder zu kommunizieren, dies auch können. Gleiches gilt für die Flags `ldap` (OpenLDAP-Support) und `mysql` (MySQL-Unterstützung). `xml` aktiviert den Support für XML-Dateien, was bei einem Webserver-System sinnvoll ist.

[7] Kapitel 5.1 beschäftigt sich mit diesem Thema ab Seite 114 ausführlich.

Wer seine Maschine nicht als Webserver installieren möchte, trifft an dieser Stelle am besten noch keine konkrete Einstellung. Die vom Profil vorgegebene Standardkonfiguration ist in den meisten Fällen akzeptabel und lässt sich auch zu einem späteren Zeitpunkt noch jederzeit korrigieren. Dafür sollten wir dann allerdings ein tieferes Verständnis der USE-Flags erworben haben, was wir in Kapitel 5.1 tun werden.

1.10 Zeitzone setzen

Damit der Rechner stets die korrekte lokale Uhrzeit anzeigt, wählt man eine passende Zeitzonendatei unterhalb von `/usr/share/zoneinfo` und kopiert sie nach `/etc/localtime`, für Deutschland etwa:

```
livecd / # date
Sat Nov 3 13:10:00 Local time zone must be set--see zic manual page 2008
livecd / # cp /usr/share/zoneinfo/Europe/Berlin /etc/localtime
livecd / # date
Sat Nov 3 13:10:05 CET 2007
```

Beim ersten Aufruf von `date` beschwert sich der Befehl noch, weil keine `localtime`-Datei zu finden ist. Beim zweiten Aufruf zeigt er dann die korrekte Zeit in CET (*Central European Time*). Sollte hier `CEST` angezeigt werden: Keine Sorge, das ist die Sommerzeit (*Central European Summer Time*).

1.11 Kernel auswählen und kompilieren

Kommen wir zum Herzstück des Systems, dem Betriebssystemkern: Gentoo bietet verschiedene Pakete auf Basis des Linux-Kernels an, die jeweils mit mehr oder minder vielen Patches versehen sind.

Den Original-Kernel, wie er unter `http://www.kernel.org/` verfügbar ist, enthält das Paket `sys-kernel/vanilla-sources`. Standard unter Gentoo ist jedoch eine gepatchte Variante, die einige Bugs und Sicherheitsprobleme beseitigt. Das Paket befindet sich als `sys-kernel/gentoo-sources` im Portage-Baum. Für Nutzer mit erhöhtem Sicherheitsbedürfnis bzw. für Server bietet sich alternativ das Paket `sys-kernel/hardened-sources` an. Einen genaueren Überblick über die in der `sys-kernel`-Kategorie verfügbaren Varianten liefert der *Gentoo Kernel Guide*, der sowohl auf Englisch[8] als auch auf Deutsch[9] zur Verfügung steht.

Wir wollen an dieser Stelle zunächst einmal absolut sicher gehen, dass wir einen gut funktionierenden Kernel installieren. Wenn unsere Maschine er-

[8] `http://www.gentoo.org/doc/en/gentoo-kernel.xml`
[9] `http://www.gentoo.org/doc/de/gentoo-kernel.xml`

folgreich von der LiveDVD gebootet hat, wissen wir bereits, dass der darauf enthaltene Kernel einwandfrei funktioniert. Der Kernel der LiveDVD entspricht dem Quellpaket `sys-kernel/gentoo-sources`, und folglich installieren wir dieses:

```
livecd / # emerge -av sys-kernel/gentoo-sources

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] sys-kernel/gentoo-sources-2.6.19-r5  USE="-build
-symlink" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
>>> sys-kernel/gentoo-sources-2.6.19-r5 merged.
>>> Recording sys-kernel/gentoo-sources in ''world'' favorites file...

>>> No packages selected for removal by clean
>>> Auto-cleaning packages...

>>> No outdated packages were found on your system.
 * GNU info directory index is up-to-date.
```

Wir haben im obigen Beispiel einmal den Abschluss einer erfolgreichen Installation mit aufgenommen. In der Hoffnung, dass `emerge` immer mit diesen Zeilen abschließt, werden wir darauf aber bei folgenden Paketinstallationen verzichten.

Natürlich kann man den Linux-Kernel wunderbar konfigurieren und optimieren, aber da er sich zu einem späteren Zeitpunkt einfach durch eine optimierte Variante austauschen lässt, ist bei einer Erstinstallation der einfachste Weg der beste.

1.11.1 Automatisiertes Kernel-Bauen mit genkernel

Am einfachsten kommt man über das Tool `genkernel` zu einem lauffähigen Kernel für das neu zu installierende System. Es konfiguriert und kompiliert die Kernel-Quellen. Da es nicht per Default installiert ist, müssen wir es an dieser Stelle einspielen:

```
livecd / # emerge -av sys-kernel/genkernel

These are the packages that would be merged, in order:

Calculating dependencies... done!
```

```
[ebuild  N    ] sys-kernel/genkernel-3.4.8  USE="-bash-completion (-ibm)
 (-selinux)" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
```

Um nun wirklich den einfachsten Weg zu wählen, machen wir uns zu Nutze, dass der Kernel der LiveDVD auf dem gerade zu installierenden System funktioniert. Ließ sich der Rechner erfolgreich von der Boot-DVD starten, muss die Konfiguration des Kernels auf der DVD für das entsprechende System passen. Entsprechend können wir die Konfiguration des aktuell laufenden Kernels aus `/proc/config.gz` für `genkernel` unbesehen übernehmen und so die Tücken einer vollständigen Kernel-Konfiguration erst einmal umgehen:

```
livecd / # zcat /proc/config.gz > \
/usr/share/genkernel/x86/kernel-config-2.6
```

Hinweis für 64-Bit-Systeme

Während Gentoo eine 64-Bit-Installation aus historischen Gründen mit `amd64` bezeichnet (AMD war die erste Firma mit gemischten 32-Bit/64-Bit-Prozessoren), identifiziert der Kernel diese mit `x86_64`, und entsprechend muss die `genkernel`-Konfiguration für eine 64-Bit-Maschine folgendermaßen angelegt werden:

```
livecd / # zcat /proc/config.gz > \
/usr/share/genkernel/x86_64/kernel-config-2.6
```

Mit dieser Basiskonfiguration ist `genkernel` in der Lage, den Kernel automatisiert zu übersetzen und zu installieren:

```
livecd / # genkernel all
* Gentoo Linux Genkernel; Version 3.4.8
* Running with options: all

* Linux Kernel 2.6.19-gentoo-r5 for x86...
* kernel: >> Running mrproper...
* config: Using config from /usr/share/genkernel/x86/kernel-config-2.6
...
* Kernel compiled successfully!
...
```

Folgt man diesem Ansatz, erhält man einen Kernel mit einer sehr breiten Hardware-Unterstützung. Für die meisten Systeme reicht ein deutlich schlankerer Kernel aus.

Auch wenn Linux-Experten den Kernel sicherlich eigenhändig konfigurieren werden, kann `genkernel` dabei helfen, einen funktionierenden Kernel zu verschlanken. Entsprechende Hinweise finden Sie im Kapitel 2.1 ab Seite 62.

1.12 Dateibaum erstellen

Wie unter Linux üblich, baut man auch bei Gentoo den Dateibaum mit Hilfe der Datei `/etc/fstab` auf. Das Handbuch vermittelt hierzu das notwendige Basiswissen, und da es keinen Unterschied zu anderen Distributionen gibt, beschränken wir uns hier darauf, die Root-Partition `/dev/hda3` unter `/` und die Boot-Partition `/dev/hda1` unter `/boot` einzuhängen, den Swap-Bereich zu aktivieren sowie das spezielle Dateisystem `shm` (für Speicherbereiche, die von mehreren Prozessen geteilt werden können) zu mounten. Außerdem binden wir auch noch das CD/DVD-Laufwerk ein und erlauben auch unprivilegierten Nutzern das Mounten einer CD oder DVD.

Die Datei editieren wir wieder mit `nano`; sie sollte zuletzt wie folgt aussehen:

```
livecd / # cat /etc/fstab
/dev/hda1   /boot        ext3    noauto,noatime       0 2
/dev/hda3   /            ext3    noatime              0 1
/dev/hda2   none         swap    sw                   0 0

shm         /dev/shm     tmpfs   nodev,nosuid,noexec  0 0

/dev/cdrom  /mnt/cdrom   auto    noauto,user          0 0
```

Falls die eigene Partitionierung von dem in Abschnitt 1.3 auf Seite 28 angegebenen Layout abweicht und/oder weitere Geräte im System vorhanden sind, sollte die Ausgabe natürlich entsprechend anders aussehen.

Wir verknüpfen hier jeweils eine der eingerichteten Partitionen (erste Spalte) mit einer bestimmten Position im Dateibaum (zweite Spalte). So liefert `/dev/hda3` die Wurzel des Dateibaumes: `/`. In der dritten Spalte führen wir jeweils das Dateisystem der Partition an.

In der vierten Spalten folgen ein paar Optionen für den `mount`-Befehl. Hier nur ein kurzer Überblick über die oben verwendeten Begriffe:

`noatime`
: Das Dateisystem soll nicht aufzeichnen, wann auf eine Datei zuge-

griffen wurde (Erstellungsdatum und Zeitpunkt der letzten Modifikation werden dagegen protokolliert).

`noauto`
: Das System wird die Partition nicht beim Booten einbinden, sondern nur wenn der Benutzer dies explizit mit `mount` anfordert.

`sw`
: Zeigt an, dass es sich um eine Swap-Partition handelt.

`nodev`
: Auf dieser Partition darf es keine Gerätedateien geben.

`nosiud`
: Auf dieser Partition darf es keine SetUID-Programme geben.

`noexec`
: Auf dieser Partition darf es keine ausführbaren Programme geben.

`user`
: Die Partition darf auch von normalen Benutzern mit Hilfe von `mount` in das System eingebunden werden.

Die fünfte Spalte spielt derzeit keine Rolle und wir können sie grundsätzlich auf `0` setzen.

Über die sechste Spalte wird festgelegt, ob und wann die Integrität des Dateisystems mit dem Prüfprogramm `fsck` überprüft werden soll. Für spezielle Dateisysteme wie `/dev/hda2` (Swap), `/dev/cdrom` oder `shm` ist eine solche Überprüfung nicht notwendig, und die sechste Spalte erhält den Wert `0`.

Mit `1` versieht man dagegen das Root-Dateisystem (`/`, hier `/dev/hda3`). Diese Partition wird dann beim Boot-Vorgang als erste auf Integrität geprüft. Alle anderen Partitionen erhalten die 2 und werden damit von `fsck` zuletzt überprüft.

Es würde an dieser Stelle zu weit führen, weiter in Einzelheiten zu gehen. Die oben angegebenen Standardwerte sind grundsätzlich zu empfehlen. Hat man weitere Partitionen definiert, dann sollte man sich bei diesen, bis auf den Wert in der sechsten Spalte, an den Angaben der Root-Partition (`/`) orientieren.

1.13 Netzwerk einrichten

Unabhängig davon, ob das System der LiveDVD schon eine Verbindung zum Netzwerk hat oder nicht, ist das neu installierte System noch nicht für die Verbindung zum Internet konfiguriert.

Hier sind nur wenige Schritte notwendig, damit auch das neue System in der gleichen Weise wie die LiveDVD das Netzwerk automatisch detektieren kann. Außerdem ist es sinnvoll, für die fertig eingerichtete Maschine den Hostnamen festzulegen. Für die komplizierteren Fälle der Netzwerkkonfiguration sei wieder auf das Kapitel 3 ab Seite 77 verwiesen. Wer seine Maschine nicht im Netzwerk betreiben möchte, kann den folgenden Abschnitt dagegen überspringen.

1.13.1 Rechnernamen festlegen

Um den Rechner in einem Netzwerk zu betreiben, muss man zumindest den Rechnernamen festlegen. Dieser findet sich in der Datei `/etc/conf.d/hostname` wieder. Wir bearbeiten diese mit `nano /etc/conf.d/hostname` und wählen hier den Namen `gentoo`, so dass der Inhalt der Datei schließlich so aussieht:

```
livecd / # cat /etc/conf.d/hostname
HOSTNAME="gentoo"
```

Vergibt ein DHCP-Server im internen Netzwerk die IP-Adressen, ist keine weitere Konfiguration notwendig, allerdings müssen wir den DHCP-Client nachinstallieren:

```
livecd / # emerge -av net-misc/dhcpcd

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] net-misc/dhcpcd-2.0.5-r1  USE="-build -debug -static" 0
kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
```

Portage sorgt in diesem Fall von sich aus dafür, dass er gleich beim Booten des Systems genutzt wird, um eine IP-Adresse zu beziehen.

Es empfiehlt sich jedoch, den Hostnamen (mit und ohne Domänenangabe) für `127.0.0.1` (eigene Adresse in `IPv4`) sowie `::1` (eigene Adresse in `IPv6`) in die Datei `/etc/hosts` einzutragen:

```
livecd / # nano /etc/hosts
```

Nach der Bearbeitung sollte die Datei dann ungefähr so aussehen:

```
livecd / # cat /etc/hosts
127.0.0.1     gentoo.example.net gentoo localhost
::1           gentoo.example.net gentoo localhost
```

Auf diese Weise sind wir in der Lage, uns auch dann mit unserem eigenen System zu verbinden (z. B. um den Apache-Webserver zu testen), wenn uns kein richtiges Netzwerk und kein externer Nameserver zur Verfügung steht.

1.13.2 Netzwerkinitialisierung beim Booten

Die Skripte innerhalb des Verzeichnisses `/etc/init.d`, die die Netzwerkverbindung beim Systemstart initialisieren, tragen per Konvention den Namen `/etc/init.d/net.`*`interface`*. Im Normalfall existieren nur `/etc/init.d/net.lo` für das Loopback-Interface und `/etc/init.d/net.eth0` für das Standardnetzwerkinterface. Während `/etc/init.d/net.lo` ein echtes Bash-Skript ist, verweist `/etc/init.d/net.eth0` als Link auf `/etc/init.d/net.lo`. Die zuletzt genannte Datei dient als zentrales Skript für die Netzwerkbehandlung; alle anderen Schnittstellen leiten sich von `net.lo` ab. Das zu behandelnde Interface entnimmt das Skript aus dem Namen des Links. Wir gehen auch im Kapitel 3 noch einmal detaillierter auf diese Mechanismen ein.

Um eine Netzwerkschnittstelle (oder mehrere) – sofern sie denn in unserem Rechner existiert – beim Boot-Vorgang automatisch in Betrieb zu nehmen, fügt man sie mit Hilfe von `rc-update` und der Option `add` zur Start-Sequenz hinzu:

```
livecd / # rc-update add net.eth0 default
 * net.eth0 added to runlevel default
```

Die zweite Option `net.eth0` benennt das zu startende Skript, und `default` bezeichnet den Standard-Boot-Vorgang. Genauer beschäftigen wir uns mit `rc-update` und dem Init-System in Kapitel 7 ab Seite 165.

1.14 Administrator-Passwort setzen

Die Installation nähert sich dem Ende – höchste Zeit, das Root-Passwort zu setzen. Hierzu nutzt man einfach das von Passwortänderungen auf anderen Distributionen bekannte Programm `passwd`:

```
livecd / # passwd
New UNIX password: geheimes_rootpasswort
Retype new UNIX password: geheimes_rootpasswort
passwd: password updated successfully
```

1.15 /etc/rc.conf anpassen

Viel Konfiguration fehlt nun nicht mehr, bis wir unser neues System das erste Mal starten können. Einige allgemeine Einstellungen vor allem in der Datei `/etc/rc.conf` sind noch notwendig, bevor wir die letzten Pakete installieren und schließlich rebooten können.

1.15.1 Standardeditor wählen

Wenn Programme von sich aus einen externen Texteditor aufrufen, startet bei Gentoo standardmäßig `nano`. Nutzer, die eher `vim` oder `emacs` gewöhnt sind, wollen dieses Verhalten in der Regel ändern – und zwar in der Datei `/etc/rc.conf`:

```
# Set EDITOR to your preferred editor.
# You may use something other than what is listed here.

EDITOR="/bin/nano"
#EDITOR="/usr/bin/vim"
#EDITOR="/usr/bin/emacs"
```

Allerdings sind sowohl `vim` als auch `emacs` nicht standardmäßig installiert und müssen hier erst mit

```
livecd / # USE="livecd" emerge -av app-editors/vim

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] dev-util/ctags-5.5.4-r2  0 kB
[ebuild  N    ] app-admin/eselect-1.0.7  USE="-bash-completion -doc"
0 kB
[ebuild  N    ] app-admin/eselect-vi-1.1.4  0 kB
[ebuild  N    ] app-editors/vim-core-7.0.174  USE="acl livecd nls
-bash-completion" 0 kB
[ebuild  N    ] app-editors/vim-7.0.174  USE="acl gpm nls perl python
-bash-completion -cscope -minimal -ruby -vim-pager -vim-with-x" 0 kB

Total: 5 packages (5 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
...
```

oder

```
livecd / # emerge -av app-editors/emacs
```

```
These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-editors/emacs-21.4-r4  USE="nls -X -Xaw3d -leim
-lesstif -motif -nosendmail" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
...
```

nachinstalliert werden, bevor man sie in der Datei `/etc/rc.conf` aktiviert. Die Angabe `USE="livecd"` bei der `vim`-Installation ist nur bei der netzwerklosen Installation notwendig, da `emerge` ansonsten ein Quellarchiv aus dem Internet herunterladen muss.

Wer sich hier schon seinen Lieblings-Editor installiert, kann natürlich im Folgenden diesen statt `nano` verwenden. Wir werden der Einfachheit halber aber weiterhin davon ausgehen, dass alle Dateien mit `nano` editiert werden.

1.16 Lokalisierung

Für den deutschen Sprachraum empfiehlt es sich, wenn möglich Unicode zu verwenden. Für die Unterstützung von Unicode auf der Kommandozeile ist die Konfiguration in der Datei `/etc/rc.conf` zuständig, die den Unicode-Support aber mittlerweile standardmäßig aktiviert:

```
# UNICODE specifies whether you want to have UNICODE support in the
# console. If you set to yes, please make sure to set a UNICODE aware
# CONSOLEFONT and KEYMAP in the /etc/conf.d/consolefont and
# /etc/conf.d/keymaps config files.

UNICODE="yes"
```

Wer eine deutsche Tastatur benutzt, die sich dem Tastenaufdruck entsprechend verhalten soll, greift wieder zum Editor `nano`, um den Eintrag `KEYMAP` in `/etc/conf.d/keymaps` zu `de-latin1` oder `de-latin1-nodeadkeys` zu korrigieren:

```
livecd / # cat /etc/conf.d/keymaps
KEYMAP="de-latin1-nodeadkeys"
SET_WINDOWKEYS="no"
EXTENDED_KEYMAPS=""
DUMPKEYS_CHARSET=""
```

Die zweite Möglichkeit (`de-latin1-nodeadkeys`) deaktiviert „tote“ Tasten, bei denen ein Tastendruck erst in Kombination mit einer zweiten Taste zu einem Zeichen zusammengesetzt wird. Mit ihnen kann man z. B. à als Kombination aus `[']` und `[A]` schreiben.

Die anderen Einstellungen bedürfen für eine einfache Tastatur keine weitere Modifikation und werden im Abschnitt 8.2.1 ab Seite 183 detaillierter erläutert.

Um das System weiter an den eigenen Sprachraum anzupassen, muss man einen für das dort verwendete Alphabet passenden Zeichensatz und die ergänzenden Formatinformationen auswählen. Als Lokalisierung bezeichnet man eben diese Kombination aus den Konventionen z. B. bei Zahl-, Datums- oder Währungsformaten mit einem passenden Zeichensatz.

Die vom System unterstützten Lokalisierungen bestimmt die Datei `/etc/locale.gen`. Sie enthält Kombinationen aus Formatdefinitionen und einem zugehörigen Zeichensatz. Für den deutschen Sprachraum eignen sich ISO-8859-1 (ohne Euro-Zeichen), ISO-8859-15 (mit Euro-Zeichen) und UTF-8:

```
de_DE ISO-8859-1
de_DE@euro ISO-8859-15
de_DE.UTF-8 UTF-8
en_US ISO-8859-1
en_US.UTF-8 UTF-8
```

Hier verknüpfen wir in jeder Zeile Lokalisierungsinformationen (z. B. `de_DE`) mit einem Zeichensatz (z. B. `UTF-8`). Die möglichen Kombinationen, die man hier eintragen kann, finden sich in `/usr/share/i18n/SUPPORTED`. Wie aus dieser Datei ersichtlich, gibt es nochmals eigene Lokalisierungsinformationen sowohl für die Schweiz (`de_CH`) als auch Österreich (`de_AT`). Genaueres hierzu findet sich aber auch noch einmal in einem eigenen Kapitel zur Lokalisierung ab Seite 186.

Aufbauend auf dieser Datei, generiert der Befehl `locale-gen` die notwendigen Zeichensätze für das System, und wir sollten das Skript, nachdem wir die fünf oben angegebenen Zeilen mit `nano` hinzugefügt haben, jetzt einmal aufrufen:

```
livecd / # locale-gen
 * Generating 5 locales (this might take a while) with 1 jobs
 *  (1/5) Generating en_US.ISO-8859-1 ...              [ ok ]
 *  (2/5) Generating en_US.UTF-8 ...                   [ ok ]
 *  (3/5) Generating de_DE.ISO-8859-1 ...              [ ok ]
 *  (4/5) Generating de_DE.ISO-8859-15@euro ...        [ ok ]
 *  (5/5) Generating de_DE.UTF-8 ...                   [ ok ]
 * Generation complete
```

Damit nun auch alle Programme die Unicode-Kodierung verwenden, muss man die entsprechenden Umgebungsvariablen setzen. Diese kann man für die Lokalisierung in `/etc/env.d/02locale` ablegen. Die Datei existiert nicht und wir können sie mit `nano /etc/env.d/02locale` anlegen. Sie sollte folgenden Inhalt haben:

```
livecd / # cat /etc/env.d/02locale
LANG="de_DE.utf8"
LC_ALL="de_DE.utf8"
```

Testet man an dieser Stelle einmal die Fehlermeldung, die man erhält, wenn man sich eine nicht existierende Datei ansehen will, merkt man, dass die Fehlermeldung noch in Englisch erscheint:

```
livecd / # ls /fehlt
ls: cannot access /fehlt: No such file or directory
```

Wie bereits erwähnt, aktualisiert `env-update` erst nach einer Veränderung an den Dateien unter `/etc/env.d` die Datei `/etc/profile`, so dass die neuen Einstellungen in neu gestarteten Shells zur Verfügung stehen. Für die aktuelle Arbeitsshell muss man sie explizit über `source /etc/profile` nachladen:

```
livecd / # env-update
livecd / # source /etc/profile
```

Jetzt redet das System auch bereitwillig in unserer Muttersprache mit uns:

```
livecd / # ls /fehlt
ls: Zugriff auf /fehlt nicht möglich: Datei oder Verzeichnis nicht
gefunden
```

1.17 Letzte Pakete einspielen

Einige wenige System-Werkzeuge fehlen an dieser Stelle noch, um nach dem Booten ein voll funktionsfähiges System zu haben. Zum einen benötigen wir ein Tool für die Protokollierung von System-Nachrichten, den so genannten *Logger*.

Hier hat man die Auswahl zwischen `app-admin/sysklogd`, `app-admin/syslog-ng` und `app-admin/metalog`. Vom Standard `app-admin/syslog-ng` abzuweichen empfiehlt sich im Wesentlichen jenen, die schon Erfahrung mit einem anderen Logger haben. Um `app-admin/syslog-ng` zu installieren und den Logger beim Starten des Systems zu initialisieren, sind folgende Befehle notwendig:

```
livecd / # emerge -av app-admin/syslog-ng

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] dev-libs/libol-0.3.18  0 kB
[ebuild  N    ] app-admin/syslog-ng-1.6.11-r1  USE="tcpd -hardened
(-selinux) -static" 0 kB

Total: 2 packages (2 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
livecd / # rc-update add syslog-ng default
 * syslog-ng added to runlevel default
```

An dieser Stelle installieren wir außerdem ein *Cron*-System, mit dem sich zeitabhängig Prozesse anstoßen (z. B. System-Checks durchführen, Logdateien archivieren etc.) lassen. Wer sich sicher ist, es nicht zu benötigen, kann diesen Teil überspringen.

Gentoo bietet wie bei den Loggern verschiedene Cron-Systeme an, die alle ähnliche Funktionalität liefern, darunter `sys-process/dcron`, `sys-process/fcron` oder `sys-process/vixie-cron`. Die unterschiedlichen Implementationen nehmen sich nicht viel; standardmäßig empfiehlt das Gentoo-Projekt `vixie-cron`:

```
livecd / # emerge -av sys-process/vixie-cron

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-mail/mailbase-1  USE="pam" 0 kB
[ebuild  N    ] sys-process/cronbase-0.3.2  0 kB
[ebuild  N    ] mail-mta/ssmtp-2.61-r2  USE="ipv6 ssl -mailwrapper
-md5sum" 0 kB
[ebuild  N    ] sys-process/vixie-cron-4.1-r10  USE="pam -debug
(-selinux)" 0 kB

Total: 4 packages (4 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
livecd / # rc-update add vixie-cron default
 * vixie-cron added to runlevel default
```

Auch der Cron-Daemon sollte beim Booten gestartet werden. Wie man ihn genau nutzt, beschreiben wir in Kapitel 16.6 ab Seite 355.

Wer auf den Festplatten-Partitionen andere Dateisysteme als `ext2` oder `ext3` verwendet muss an dieser Stelle auch noch die entsprechenden Werk-

zeuge für diese Dateisysteme installieren. Die entsprechenden Pakete befinden sich in der Kategorie `sys-fs`. So benötigt z. B. XFS das Paket `sys-fs/xfsprogs` und JFS `sys-fs/jfsutils`.

Das Gentoo-Benutzerhandbuch beschreibt an dieser Stelle auch das Anlegen von Benutzern; wir wollen aber erst einmal als `root` weiter arbeiten und dieses allgemeine Thema der Linux-Administration nicht weiter ausführen.

Dass wir hier weiterhin das Administrator-Konto verwenden sollten Sie allerdings als Ausnahme werten: Bei normaler Nutzung des Systems sollte man sich grundsätzlich als normaler Benutzer einloggen. Für unsere weiteren Erklärungen wäre es allerdings zu umständlich, bei jedem Kommando gesondert anzugeben, dass Sie dafür Administrator-Rechte benötigen. Schließlich wollen wir hier gerade die Administration des Systems erklären.

1.17.1 Fertig machen zum Booten

Fehlt nur noch der Boot-Loader: Als Standard empfiehlt Gentoo GRUB, den wir wie folgt installieren:

```
livecd / # emerge -av sys-boot/grub

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] sys-boot/grub-0.97-r3  USE="-custom-cflags -netboot
-static" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
```

Die Konfigurationsdatei legt man mit `nano` unter `/boot/grub/grub.conf` an, und sie sollte in der Minimalkonfiguration so aussehen:

```
livecd / # cat /boot/grub/grub.conf
default 0
timeout 30

title=Gentoo Linux
root (hd0,0)
kernel /kernel-genkernel-x86-2.6.19-gentoo-r5 root=/dev/ram0 init=/linu\
xrc ramdisk=8192 real_root=/dev/hda3 udev
initrd /initramfs-genkernel-x86-2.6.19-gentoo-r5
```

Dabei muss die Kernel-Bezeichnung dem Namen der mit `genkernel` installierten Kernel-Datei entsprechen. Um den korrekten Dateinamen herauszufinden, wirft man einen Blick ins Verzeichnis `/boot`:

```
livecd / # ls /boot
System.map-genkernel-x86-2.6.19-gentoo-r5
initramfs-genkernel-x86-2.6.19-gentoo-r5
grub
kernel-genkernel-x86-2.6.19-gentoo-r5
```

Die aufgezeigte Konfiguration orientiert sich am auf Seite 28 erstellten einfachen Partitionierungsschema, das `/dev/hda1` als `/boot` und `/dev/hda3` als Root-Partition verwendet. `/boot` machen wir mit `root (hd0,0)` zur Basispartition von GRUB und geben den Dateinamen des Kernels demnach relativ zu `/boot` an. Entsprechend heißt es *nicht* `/boot/kernel-`*`version`*, sondern nur `/kernel-`*`version`*. Die Root-Partition übergeben wir dem Kernel beim Start als Parameter `real_root=/dev/hda3`.

Die übrigen Parameter für den Kernel (`root=/dev/ram0`, `init=/linuxrc`, `ramdisk=8192` und `udev`) erlauben es, sich über ein kleines Hilfssystem, die *initiale Ramdisk*, in das eigentliche System zu hangeln. Die initiale Ramdisk liegt ebenfalls im Boot-Verzeichnis als `initramfs-genkernel-x86-2.6.19-gentoo-r5` und wird mit der Option `initrd` in der `grub.conf` für den Boot-Prozess verfügbar gemacht und beim Booten in den Speicher geladen.

Damit der Kernel die Datei auch wirklich verwendet, müssen wir ihm mit `root=/dev/ram0` angeben, dass er zunächst auf Basis der Ramdisk booten soll. `ramdisk=8192` bezeichnet dabei die Größe der Datei. Die Angabe `init=/linuxrc` verweist auf eine in der Ramdisk abgelegte Datei, die sich um den Systemstart auf Basis der Ramdisk kümmert und dafür sorgt, dass zusätzliche Module für die Hardware-Unterstützung geladen werden. `udev` gibt darüber hinaus an, dass die Ramdisk auf Basis der dynamischen Geräteverwaltung `udev` arbeiten soll, so dass die Geräte des Systems wenn möglich automatisch erkannt werden.

Diese hier angegebene einfache GRUB-Konfiguration aktiviert keine grafischen Features wie z. B. ein Hintergrundbild, das das System beim Booten zeigt, die man auf Server-Systemen ohnehin nicht braucht. Wir gehen im nächsten Kapitel unter 2.1.7 ab Seite 71 kurz darauf ein, aber für komplexere Konfigurationen sollte man die GRUB-Dokumentation zu Rate ziehen.[10]

Wer ein Dual-Boot-System mit Windows installieren möchte, muss an dieser Stelle an die im Anhang unter B.2 beschriebene zusätzliche GRUB-Konfiguration zurückkehren.

[10] Eine ausführliche Erklärung zur GRUB-Konfiguration bietet auch Anke Börnig in ihrem ebenfalls bei Open Source Press erschienenen Buch „LPIC-2“, ISBN 978-3-937514-20-8.

GRUB erlaubt es, die angebotenen Boot-Varianten einfach über das Boot-Menü zu editieren. So kann man bei Bedarf die übergebenen Kernel-Parameter oder auch den zu startenden Runlevel (siehe Kapitel 7 ab Seite 165) ändern, indem man über die Taste [E] in den Editiermodus wechselt. Das hilft einerseits bei Boot-Problemen, ebnet aber auch einem böswilligen Angreifer den Weg ins System. Wer sich hier absichern möchte, fügt im Kopf der GRUB-Konfiguration die password-Option hinzu:

```
password mehrsicherheit
```

grub wird das angegebene Passwort dann beim Wechsel in den Editiermodus abfragen.

Damit bleibt abschließend nur noch, den neuen Boot-Sektor zu schreiben. Dafür muss GRUB wissen, welche Partitionen derzeit gemountet sind. Diese Information ist im Chroot-System nicht verfügbar, man kann sie aber problemlos zugänglich machen, indem man sie aus /proc/mounts ausliest und in die Datei /etc/mtab schreibt:

```
livecd / # grep -v rootfs /proc/mounts > /etc/mtab
```

Den Boot-Sektor schreibt nun der folgende Befehl:

```
livecd / # grub-install --no-floppy /dev/hda
Probing devices to guess BIOS drives. This may take a long time.
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script ‘grub-install’.

(hd0)/dev/hda
```

Hier hält die Option --no-floppy den Befehl grub-install davon ab, Zeit für die Suche nach einem Diskettenlaufwerk zu verschwenden.

Damit können wir die chroot-Umgebung verlassen; der neue Server sollte sich nun booten lassen:

```
livecd / # exit
livecd ~ # reboot
```

Vergessen Sie nicht, die LiveDVD aus dem Laufwerk zu nehmen, damit Ihr Rechner auch wirklich von der Festplatte bootet.

Kapitel 2

Der Kernel

An dieser Stelle haben wir hoffentlich zum ersten Mal erfolgreich das neue System gebootet und können uns als `root` mit dem im vorigen Kapitel festgelegten Passwort einloggen. Anschließend erwartet Gentoo unsere Eingaben auf der Kommandozeile:

```
gentoo ~ #
```

Das System ist nun noch nicht sonderlich üppig ausgestattet; um weitere Pakete zu installieren, nutzen wir, wenn möglich, eine Netzwerkverbindung, wobei der Kernel vorhandene Netzwerkkarten unterstützen muss – und dies meist auch automatisch tut. Bei ausgefallenerer Hardware werden wir um eine manuelle Konfiguration aber nicht herum kommen.

Auf der anderen Seite haben wir während der Installation einen Kernel mit sehr breiter Hardware-Unterstützung erstellt, so dass wir unser Betriebssystem etwas entschlacken und die Unterstützung nicht vorhandener Hardware deaktivieren.

Steht eine Netzwerkverbindung bereits zur Verfügung, so sei empfohlen, zum nächsten Kapitel zu springen und Veränderungen am Kernel hier zu unterlassen. Es gibt wichtigere Stellschrauben in einem Gentoo-System, und wir können jederzeit an diese Stelle zurückkehren, wenn wir ein wenig Erfahrung mit dem System gesammelt haben.

Ein korrekt konfigurierter Kernel, der die vorhandene Hardware vollständig unterstützt, ist essentiell für ein funktionierenden Linux-System. Der Experte wird sich mit der Konfiguration des Kernels manuell auseinander setzen wollen, aber als Anfänger verliert man sich sehr leicht in der Komplexität dieses Unterfangens und endet schnell mit einem defekten System. Gentoo bietet zur Unterstützung des Kernel-Baus das Werkzeug `genkernel`, mit dem sich dieses Kapitel näher befasst.

2.1 genkernel

Da die Kernel-Konfiguration schon fast eine Wissenschaft für sich ist, gehen wir an dieser Stelle nicht auf Details der Kernel-Parameter ein. Stattdessen beschränken wir uns darauf, auf die Netzwerkkonfiguration hinzuweisen und den anfänglichen, auf der Konfiguration der LiveDVD basierenden Kernel zu entschlacken, um `genkernel` dabei im Detail kennen zu lernen.

Es geht hier keineswegs darum, mit dem Kernel herumzuspielen, auf dass man zuletzt mit einem defekten System endet. Gerade bei Experimenten mit dem Kernel kann so etwas schnell passieren. Wer also glücklich über sein gerade gebootetes Gentoo ist und wenig Lust verspürt, sich mit den Untiefen der Kernel-Konfiguration auseinander zu setzen, sollte den nachfolgenden Teil als Überblick über `genkernel` betrachten und vor allem die verzichtbaren Elemente ab Seite 66 überspringen.

Wie man einen lauffähigen Kernel aus einer funktionierenden LiveDVD generiert, haben wir in Kapitel 1.11.1 ab Seite 47 gezeigt. Von dieser Konfiguration gehen wir im Folgenden aus. `genkernel` übernimmt allerdings nur das Übersetzen der Kernel-Quellen und die korrekte Installation des fertigen Kernel-Produkts. Bei der richtigen Wahl der Kernel-Parameter hilft es nicht.

2.1.1 Die grundlegenden Funktionen

Wir müssen `genkernel` unter Nennung einer durchzuführenden Aktion aufrufen. Wie im Kapitel 1.11.1 gesehen, lautet die einfachste, aber gleichzeitig umfangreichste Handlung `all`:

```
gentoo ~ # genkernel all
```

Dieser Befehl bewirkt, dass `genkernel` das initiale Dateisystem aus der Initial RAM-Disk (*Initrd*), den Kernel selbst und schließlich die Kernel-Module erstellt und installiert. Jede dieser Aktionen lässt sich auch einzeln ausführen. `genkernel initrd` erstellt nur das initiale Dateisystem, welches der Kernel beim Booten quasi als Vorstufe des Dateisystems verwendet, wenn die eigentliche Festplatte noch nicht eingebunden wurde. `genkernel bzImage` erstellt ausschließlich den Kernel, und `genkernel kernel` generiert sowohl den Kernel als auch die Module. Die meisten Nutzer benötigen jedoch nicht mehr als `genkernel all`.

Bevor wir einen neuen Kernel erstellen, sollten wir zusehen, dass wir diesen auch wirklich in unserem Boot-Verzeichnis installieren können. Wir hatten während der Installation auf Seite 50 mit der `mount`-Option `noauto` festgelegt, dass das `/boot`-Verzeichnis nicht automatisch während des Rechnerstarts eingebunden wird. Wir holen das hier nach, indem wir `mount` explizit aufrufen:

```
gentoo ~ # mount /boot
```

2.1.2 Die Konfiguration

Das Verhalten von `genkernel` lässt sich über eine Konfigurationsdatei beeinflussen: `/etc/genkernel.conf` ist in zwei Teile unterteilt, von denen der erste die Benutzer-Optionen umfasst und der zweite interne Variablen des Programms festlegt. Folglich werden wir nur den oberen Teil verändern. Die dort angegebenen Einstellungen lassen sich allerdings auch alle über `genkernel`-Parameter auf der Kommandozeile beeinflussen, und wir werden in den folgenden Abschnitten klären, welche Optionen äquivalent sind.

2.1.3 Den Kernel reduzieren

Bevor wir mit `genkernel` eine reduzierte Kernel-Konfiguration, ausgehend von der LiveDVD-Variante, erstellen können, müssen wir sicherstellen, dass die erste, funktionierende Konfiguration wirklich verfügbar ist. Die Standard-Konfiguration speichert der Aufruf `genkernel all` (siehe Seite 47) standardmäßig unter `/etc/kernels/` ab:

```
gentoo ~ # ls /etc/kernels
kernel-config-x86-2.6.19-gentoo-r5
```

Ob dies tatsächlich geschieht, lässt sich über die Optionen `--safe-config` und `--no-safe-config` aktivieren bzw. abstellen. Den Standard gibt die Variable `SAVE_CONFIG` in der Konfigurationsdatei `/etc/genkernel.conf` vor: `SAVE_CONFIG="yes"` schaltet die Speicherung ein.

```
# Save the new configuration in /etc/kernels upon
# successfull compilation
SAVE_CONFIG="yes"
```

Bevor wir die lauffähige Kernel-Konfiguration aus Versehen überschreiben, kopieren wir diese einmal zur Sicherheit:

```
gentoo ~ # cp /etc/kernels/kernel-config-x86-2.6.19-gentoo-r5 \
/etc/kernels/kernel-config-x86-2.6.19-gentoo-r5-LiveDVD
```

Damit können wir im Notfall auf eine funktionierende Kernel-Konfiguration zurückgreifen. Um nun den Konfigurationsdialog des Kernels aufzurufen, gibt es drei verschiedene `genkernel`-Optionen:

`--menuconfig`
: ruft zur Konfiguration ein textbasiertes, pseudografisches Interface auf.

`--xconfig`
: öffnet ein grafisches Frontend, das nicht mehr verlangt als einen laufenden X-Server.

`--gconfig`
: erlaubt bei laufendem X-Server die Konfiguration über ein GTK-basiertes Frontend.

Da wir keinen X-Server voraussetzen, bietet sich nur `--menuconfig` an. Wer mag kann auch die Variable `MENUCONFIG` in der Konfigurationsdatei von `MENUCONFIG="no"` auf `"yes"` setzen und damit auf die `--menuconfig`-Option auf der Kommandozeile verzichten.

```
# Run 'make menuconfig' before compiling this kernel?
MENUCONFIG="yes"
```

Bevor wir `genkernel` nun aufrufen, sollten wir zwei Dinge sicherstellen:

- Wir wollen den alten Kernel nicht überschreiben und sollten daher dem neuen Kernel einen anderen Namen geben.
- Bereits kompilierte Teile des Kernels wollen wir, wenn möglich, nicht nochmals kompilieren.

Den Kernel versehen wir über den Parameter `--kernname` mit einem neuen Namen. Dieser ersetzt die Zeichenkette `genkernel`, die `genkernel` normalerweise in den Dateinamen der resultierenden Dateien in `/boot` einbaut

```
gentoo ~ # ls /boot
System.map-genkernel-x86-2.6.19-gentoo-r5
initramfs-genkernel-x86-2.6.19-gentoo-r5
grub
kernel-genkernel-x86-2.6.19-gentoo-r5
```

durch eine andere, z. B. `reduced`. Mit `--kernname reduced` heißt der neue Kernel also `kernel-reduced-x86-2.6.19-gentoo-r5`. Damit bleibt die alte Kernelversion erhalten, und wir haben jederzeit einen Kern zur Verfügung, der wirklich bootet.

Den zweiten Wunsch, die schon kompilierten Dateien so weit wie möglich zu erhalten, erfüllen wir uns mit der Option `--no-clean`. Normalerweise würde `genkernel` im Kernel-Quellverzeichnis `/usr/src/linux` ein `make clean` durchführen und damit bereits kompilierte Dateien löschen.

`genkernel` wird das Kernel-Quellverzeichnis automatisch mit `clean` aufräumen, da die Konfigurationsdatei standardmäßig `CLEAN="yes"` setzt. Wer mag kann diese Option auch in der Konfiguration ändern und dann auf die `--no-clean`-Option verzichten:

```
# Run 'make clean' before compilation?
# If set to NO, implies MRPROPER WILL NOT be run
# Also, if clean is NO, it won't copy over any configuration
# file, it will use what's there.
CLEAN="no"
```

Mit all diesen Parametern rufen wir `genkernel` nun folgendermaßen auf:

```
gentoo ~ # genkernel --menuconfig --kernname=reduced --no-clean all
* Gentoo Linux Genkernel; Version 3.4.8
* Running with options: --menuconfig --kernname=reduced --no-clean all

* Linux Kernel 2.6.19-gentoo-r5 for x86...
* mount: /boot mounted successfully!
* config: --no-clean is enabled; leaving the .config alone.
* config: >> Invoking menuconfig...
```

An dieser Stelle erscheint das Kernel-Konfigurationsmenü, in dem wir unnötige Subsysteme und Module deaktivieren, um danach den Kernel neu zu kompilieren.

Wir können hier die verschiedenen Kernel-Einstellungen nicht umfassend diskutieren, da die Auswahl notwendiger und verzichtbarer Optionen sehr stark von der verwendeten Hardware abhängt. Wir wollen aber zeigen, wo sich die Einstellungen für Netzwerkkarten verstecken und welche Subsysteme sich bei den meisten Rechnern komplett deaktivieren lassen, wenn die entsprechende Hardware nicht vorhanden ist.

Wer ein in Sachen Hardware standardisiertes System verwendet oder einen bestimmten Laptop-Typ sollte vor Experimenten mit dem Kernel auf jeden Fall die Hardware-Sektion des Gentoo-Wiki[1] konsultieren. Vielfach finden sich hier für verbreitete Rechnersysteme wichtige Hinweise für die Kernel-Konfiguration wie auch Tipps zur allgemeinen Konfiguration.

2.1.4 Netzwerkkarten

Es gibt zwei Sektionen, in denen sich die Treiber für Netzwerkkarten befinden können. Die meisten Treiber finden sich unter

Device Drivers | Network device support
: Die verschiedenen Treiber verstecken sich hinter den Einträgen **ARCnet**, **PHY device support**, **Ethernet (...)**, **Token Ring Devices**, **Wireless LAN**, **WAN interfaces** und **ATM devices**.

Die gebräuchlichen Karten sind meist solche aus einer der **Ethernet**-Sektionen. Hier sind auch fast alle Karten standardmäßig als Modul aktiviert und sollten vom Kernel unterstützt werden.

USB-basierte Netzwerkkarten finden sich dagegen in der USB-Sektion:

Device Drivers | USB-Support
: Hier finden sich noch einmal einige Netzwerktreiber unter den **USB Network Adapters**.

Und schließlich finden sich noch exotischere Varianten im allgemeinen **Networking**-Abschnitt:

Networking | IrDA (infrared) subsystem support
: Für Infrarotgeräte.

Networking | Bluetooth subsystem support
: Für die Unterstützung von Bluetooth-Verbindungen.

Ist die eigene Hardware hier nicht verzeichnet, bleibt eventuell die Möglichkeit, ein externes Modul zu installieren. Wir beschreiben dies in Abschnitt 2.2.1.

2.1.5 Verzichtbare Elemente

Kommen wir zu den Treibern, auf die der Nutzer im Normalfall verzichten kann und ohne die der Kernel kleiner und übersichtlicher wird.

[1] `http://gentoo-wiki.com/Index:Hardware`

Die folgenden Punkte entsprechen jeweils kompletten Subsystemen, die im Kernel der LiveDVD aktiviert sind, jedoch nur selten gebraucht werden. Vor deren Deaktivierung muss man sich natürlich vergewissern, dass die jeweiligen Systeme auch wirklich fehlen:

Bus options (PCI, PCMCIA, EISA, MCA, ISA)

PCCARD (PCMCIA/CardBus) support
Lässt sich deaktivieren, wenn kein PCMCIA-Slot vorhanden ist, was bei Nicht-Laptop-Systemen die Regel ist.

Networking

Bluetooth subsystem support
kann deaktiviert werden, wenn die Maschine keine Bluetooth-Schnittstelle besitzt.

Device Drivers

Multi-device support (RAID and LVM)
Wer ein System ohne RAID-Platte besitzt kann auf diese Module verzichten.

Fusion MPT device support
Ohne spezielle Fusion-MPT-SCSI- oder Netzwerkgeräte kann man getrost auf diese Module verzichten.

IEEE 1394 (FireWire) support
Fehlt dem Rechner die FireWire-Schnittstelle, werden auch die entsprechenden Module nicht benötigt.

I2O device support
I2O ist dem SCSI-Protokoll vergleichbar und wird nur von wenigen Geräten unterstützt. Im Normalfall kann dieser Bereich deaktiviert werden.

ISDN subsystem
Wer keine ISDN-Karte besitzt kann folglich auch diese Module aus dem System entfernen.

Speakup console speech
Wer den Output der Konsole lieber liest als hört kann auf das **Speakup**-System verzichten.

InfiniBand
Eine Form der seriellen Übertragung, die nur von wenigen Geräten genutzt wird.

Es gibt zahlreiche Gerätetreiber, die als Modul aktiviert sind und die nur in Verbindung mit dem entsprechenden Gerät benötigt werden. Wer sich also die Mühe machen möchte kann den Kernel weiter ausdünnen.

Generell sollte man jedoch Vorsicht beim Deaktivieren von Treibern walten lassen und vor allem darauf verzichten, Elemente zu deaktivieren, nur weil

einem der Name nichts sagt. Ist ein Treiber klar als spezifisches Modul für einen bestimmten Gerätetyp erkennbar, kann er deaktiviert werden. Aber gerade wenn es um Treiber mit breiterer Geräte-Unterstützung geht, sollte man sich erst einmal zurückhalten, bis man nachvollzogen hat, welche Funktion das entsprechende Modul hat.

Hat man die unnötigen Elemente deaktiviert, verlässt man das Menü über **Exit** und speichert auf Nachfrage die neue Kernel-Konfiguration. `genkernel` fährt nun mit seiner Arbeit fort:

```
*         >> Compiling 2.6.19-gentoo-r5 bzImage...
*         >> Compiling 2.6.19-gentoo-r5 modules...
* Copying config for successful build to /etc/kernels/kernel-config-x86-
2.6.19-gentoo-r5
* initramfs: >> Initializing...
*         >> Appending base_layout cpio data...
*         >> Appending auxilary cpio data...
*         >> Appending busybox cpio data...
*         >> Appending insmod cpio data...
*         >> Appending modules cpio data...
*
* Kernel compiled successfully!
*
* Required Kernel Parameters:
*     real_root=/dev/$ROOT
*
*     Where $ROOT is the device node for your root partition as the
*     one specified in /etc/fstab
*
* If you require Genkernel's hardware detection features; you MUST
* tell your bootloader to use the provided INITRAMFS file. Otherwise;
* substitute the root argument for the real_root argument if you are
* not planning to use the initrd...

* WARNING... WARNING... WARNING...
* Additional kernel cmdline arguments that *may* be required to boot pro
perly...
* add ''vga=791 splash=silent'' if you use a bootsplash framebuffer

* Do NOT report kernel bugs as genkernel bugs unless your bug
* is about the default genkernel configuration...
*
* Make sure you have the latest genkernel before reporting bugs.
```

Nach Abschluss des Vorgangs finden wir den neuen Kernel ebenfalls im Verzeichnis `/boot`:

```
gentoo ~ # ls /boot
System.map-genkernel-x86-2.6.19-gentoo-r5
System.map-reduced-x86-2.6.19-gentoo-r5
initramfs-genkernel-x86-2.6.19-gentoo-r5
```

```
initramfs-reduced-x86-2.6.19-gentoo-r5
grub
kernel-genkernel-x86-2.6.19-gentoo-r5
kernel-reduced-x86-2.6.19-gentoo-r5
```

2.1.6 Eine sichere Boot-Konfiguration

Damit der neue Kernel beim Booten auch zur Verfügung steht, müssen wir ihn zur GRUB-Konfiguration hinzufügen. `/boot/grub/grub.conf` sollte dann so aussehen:

```
default 0
timeout 30

title=Gentoo Linux (Reduced Kernel)
root (hd0,0)
kernel /kernel-reduced-x86-2.6.19-gentoo-r5 root=/dev/ram0 init=/linuxr\
c ramdisk=8192 real_root=/dev/hda3 udev
initrd /initramfs-reduced-x86-2.6.19-gentoo-r5

title=Gentoo Linux
root (hd0,0)
kernel /kernel-genkernel-x86-2.6.19-gentoo-r5 root=/dev/ram0 init=/linu\
xrc ramdisk=8192 real_root=/dev/hda3 udev
initrd /initramfs-genkernel-x86-2.6.19-gentoo-r5
```

Damit bietet GRUB beim Systemstart zwei verschiedene Kernel zur Auswahl. Diese Vorgehensweise ist grundsätzlich zu empfehlen: einen erprobten und sicher funktionierenden Kernel, und einen zweiten, den wir verändert, aber noch nicht getestet haben. So lässt sich das System auch booten, wenn der neue Kernel fehlerhaft ist.

Vereinfachen kann man sich diese Organisation durch entsprechendes Verlinken des Kernels:

```
gentoo ~ # cd /boot
gentoo boot # for fl in *genkernel*;do ln -s $fl \
> ${fl/-genkernel*/-safe};done
gentoo boot # for fl in *reduced*;do ln -s $fl ${fl/-reduced*/}; done
gentoo boot # ls -la
total 8477
drwxr-xr-x  4 root root    4096 Jan 25 11:44 .
drwxr-xr-x 21 root root     520 Nov 20 14:47 ..
lrwxrwxrwx  1 root root      39 Jan 25 11:44 System.map -> System.map-re
duced-x86-2.6.19-gentoo-r5
-rw-r--r--  1 root root  674983 Oct 12 14:04 System.map-genkernel-x86-2.
6.19-gentoo-r5
-rw-r--r--  1 root root  653889 Jan 25 10:44 System.map-reduced-x86-2.6.
```

```
19-gentoo-r5
lrwxrwxrwx  1 root root      41 Jan 25 11:44 System.map-safe -> System.m
ap-genkernel-x86-2.6.19-gentoo-r5
drwxr-xr-x  2 root root    4096 Oct 12 16:25 grub
lrwxrwxrwx  1 root root      38 Jan 25 11:44 initramfs -> initramfs-redu
ced-x86-2.6.19-gentoo-r5
-rw-r--r--  1 root root 2243032 Oct 12 15:21 initramfs-genkernel-x86-2.6
.19-gentoo-r5
-rw-r--r--  1 root root 1898019 Jan 25 11:39 initramfs-reduced-x86-2.6.1
9-gentoo-r5
lrwxrwxrwx  1 root root      40 Jan 25 11:44 initramfs-safe -> initramfs
-genkernel-x86-2.6.19-gentoo-r5
lrwxrwxrwx  1 root root      35 Jan 25 11:44 kernel -> kernel-reduced-x8
6-2.6.19-gentoo-r5
-rw-r--r--  1 root root 1600800 Oct 12 14:04 kernel-genkernel-x86-2.6.19
-gentoo-r5
-rw-r--r--  1 root root 1548843 Jan 25 10:44 kernel-reduced-x86-2.6.19-g
entoo-r5
lrwxrwxrwx  1 root root      37 Jan 25 11:44 kernel-safe -> kernel-genke
rnel-x86-2.6.19-gentoo-r5
drwx------  2 root root   16384 Oct 12 11:49 lost+found
gentoo boot # cd ~
gentoo ~ #
```

Wir erhalten dadurch die Verknüpfungen `kernel` und `kernel-safe` (sowie die zugehörigen Dateien `System.map`, `System.map-safe`, `initramfs` und `initramfs-safe`), die auf die entsprechenden Originaldateien verweisen. Damit können wir `/boot/grub/grub.conf` folgendermaßen vereinfachen:

```
default 0
timeout 30

title=Gentoo Linux
root (hd0,0)
kernel /kernel root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root=/dev\
/hda3 udev
initrd /initramfs

title=Gentoo Linux (Safe Kernel)
root (hd0,0)
kernel /kernel-safe root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root\
=/dev/hda3 udev
initrd /initramfs-safe
```

GRUB wird so den neuen Kernel booten, aber zur Sicherheit den alten ebenfalls anbieten.

Sobald wir einen neuen Kernel kompiliert haben, entfernen wir den alten, als `kernel-safe` verlinkten Kernel inklusive den zugehörigen `initrd`- und `System.map`-Dateien sowie den `*-safe`-Verknüpfungen. Danach be-

nennen wir die `kernel-` sowie die zugehörigen `initrd-` und `System.map`-Verknüpfungen in die entsprechenden `*-safe`-Varianten um und verlinken schließlich die Dateien des neuen Kernels mit den frei gewordenen `kernel-`, `initrd-` und `System.map`-Links. So müssen wir die GRUB-Konfiguration nicht jedes Mal neu anpassen.

2.1.7 Spielzeug: Splash-Screen

Die LiveDVD wechselt beim Booten recht zügig in einen grafischen Modus, der optisch deutlich ansprechender ist als die schnell vorüberhuschenden Textzeilen – im Grunde reine Kosmetik, aber `genkernel` beherrscht auch die Erstellung eines solchen *Splash-Screens*, und darum wollen wir die Konfiguration nicht verheimlichen.

Hinweis für Systeme mit Netzwerkanbindung

Die Pakete für den Splash-Screen lassen sich nur installieren, wenn wir bereits eine Netzwerkverbindung zur Verfügung und unser System mit `emerge --sync` aktualisiert haben.

Bei einer Erstinstallation können Sie diesen Abschnitt ohne Bedenken überspringen.

Die notwendigen Einstellungen sind etwas komplexer, und wir werden hier nur die wichtigsten Schritte beleuchten, um den Splash-Screen auf den meisten Systemen zum Laufen zu bekommen. Bei Schwierigkeiten oder komplexeren Konfigurationen bietet sich die entsprechende Seite im deutschen Gentoo-Wiki[2] oder dem englischen[3] an.

Über die Befehle `--gensplash` bzw. `--bootsplash` aktiviert `genkernel` für den Kernel den Support für einen Splash-Screen. Der Boot-Prozess wird damit, wie bei der LiveDVD auch, grafisch begleitet und die Vielzahl an Meldungen des Init-Systems (siehe Kapitel 7) ausgeblendet. Man kann sich hier nur für `gensplash` oder `bootsplash` entscheiden. Genaueres findet man auf der Homepage des `gensplash`-Autors[4]. Für `gensplash` muss das Paket `media-gfx/splashutils`, für `bootsplash` das Paket `media-gfx/bootsplash` installiert sein, andernfalls meldet `genkernel` während der Ausführung:

```
*       >> No splash detected; skipping!
```

[2] `http://de.gentoo-wiki.com/Fbsplash`
[3] `http://gentoo-wiki.com/HOWTO_fbsplash`
[4] `http://dev.gentoo.org/~spock/projects/gensplash`

Als Standard wählt genkernel über die Konfigurationsoption BOOTSPLASH="yes" bootsplash aus, aber da das Projekt gensplash mittlerweile weiter entwickelt ist als das ursprüngliche bootsplash, sei die Einstellung GENSPLASH="yes" empfohlen.

```
# Copy bootsplash into the initrd image?
GENSPLASH="yes"
```

Wer also den grafischen Boot-Modus bevorzugt installiert an dieser Stelle das zugehörige Paket media-gfx/splashutils.

```
gentoo ~ # USE="png mng truetype" emerge -av media-gfx/splashutils

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] media-libs/jpeg-6b-r7  0 kB
[ebuild  N    ] media-libs/libpng-1.2.15  USE="-doc" 0 kB
[ebuild  N    ] media-libs/freetype-2.1.10-r3  USE="zlib -bindist -doc"
0 kB
[ebuild  N    ] dev-lang/swig-1.3.31  USE="perl python -doc -guile -java
 -lua -mono -ocaml -php -pike -ruby -tcl -tk" 0 kB
[ebuild  N    ] dev-perl/Locale-gettext-1.05  0 kB
[ebuild  N    ] dev-libs/klibc-1.4.13  USE="-debug (-n32)" 463 kB
[ebuild  N    ] media-libs/lcms-1.15  USE="python zlib -jpeg -tiff" 0 kB
[ebuild  N    ] sys-apps/help2man-1.36.4  USE="nls" 0 kB
[ebuild  N    ] media-gfx/fbgrab-1.0  0 kB
[ebuild  NS   ] sys-devel/automake-1.9.6-r2  0 kB
[ebuild  N    ] media-libs/libmng-1.0.9-r1  USE="-lcms" 0 kB
[ebuild  N    ] media-gfx/splashutils-1.4.1  USE="gpm mng png truetype -
hardened" 1,515 kB

Total: 12 packages (11 new, 1 in new slot), Size of downloads: 1,977 kB

Would you like to merge these packages? [Yes/No] Yes
```

Hier aktivieren wir auf etwas unorthodoxem Wege (siehe Seite 124) mehrere USE-Flags und erreichen damit, dass splashutils verschiedene Bildformate und TrueType-Schriften unterstützt. USE-Flags sind ein etwas komplexeres Thema, das wir ausführlich in Kapitel 5.1 behandeln. Dort behandeln wir auch, wie wir die USE-Flags mit flagedit korrekt setzen.

Die Werkzeuge für den Splash-Screen haben wir jetzt, aber es fehlt noch ein ansprechendes grafisches Thema, das media-gfx/splashutils nicht automatisch mitliefert. Für das Thema der LiveDVD installieren wir media-gfx/splash-themes-livecd. Einige weitere Alternativen finden sich in media-gfx/splash-themes-gentoo.

```
gentoo ~ # emerge -av media-gfx/splash-themes-livecd

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] media-gfx/splash-themes-livecd-2007.0  4,649 kB

Total: 1 package (1 new), Size of downloads: 4,649 kB

Would you like to merge these packages? [Yes/No]
```

Die Themen installiert `emerge` im Verzeichnis `/etc/splash`, das Thema der LiveDVD unter `/etc/splash/livecd-2007.0`.

Jetzt rufen wir `genkernel` nochmals mit der `--gensplash`-Option auf:

```
gentoo ~ # genkernel --kernname=reduced --no-clean \
--gensplash=livecd-2007.0 all
```

Auf die Option `--menuconfig` verzichten wir, da wir die Konfiguration nicht verändern, sondern nur die für den Splash-Screen notwendigen Dateien in der `initramfs`-Datei ablegen.

In diesem Durchlauf sollte sich `genkernel` zwischendurch mit

```
*   >> Installing gensplash [ using the livecd-2007.0 theme ]...
```

melden und damit anzeigen, dass das Thema erfolgreich installiert wurde.

Sowohl bei `bootsplash` als auch bei `gensplash` braucht man neben den installierten Paketen zusätzliche Angaben in der GRUB-Konfiguration. Für `bootsplash` fügt man `vga=791 splash=silent` zu der `kernel`-Zeile in der `grub.conf` hinzu, bei `gensplash` hingegen `splash=silent,theme:THEME vga=791 CONSOLE=/dev/tty1 quiet`. `THEME` steht für das ausgewählte grafische Thema. Die `vga`-Option bezeichnet den Video-Modus (hier 1024×768 Pixel bei 24 Bit Farbtiefe), `splash=silent` startet den Splash-Screen in dem Modus, bei dem die Textzeilen des Init-Systems ausgeblendet sind. Der Modus lässt sich beim Booten über die `[F2]`-Taste wechseln.

Damit gelangen wir schließlich zu der folgenden Konfiguration für unseren Eintrag `Gentoo Linux` in der Datei `/boot/grub/grub.conf`:

```
title=Gentoo Linux
root (hd0,0)
kernel /kernel root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root=/dev\
/hda3 splash=silent,theme:livecd-2007.0 vga=791 CONSOLE=/dev/tty1 quiet\
 udev
initrd /initramfs
```

2.1.8 Weitere genkernel-Optionen

Schließen wir das Kapitel über `genkernel` mit einigen weniger bekannten Optionen, die sich bei häufigerem Gebrauch jedoch als recht nützlich erweisen.

So verhindert `--no-install`, dass `genkernel` den Kernel in `/boot` installiert. Einzelteile für den Boot-Prozess (`kernel`, `System.map`, `initramfs`) findet man dadurch in `/var/tmp/genkernel` und kann sie manuell nach `/boot` verschieben.

Wer Nutzern beim Booten die Keymap für die Unterstützung verschiedener Keyboards (wie auf der LiveDVD) auswählen lassen möchte, aktiviert dieses Feature mit `--do-keymap-auto`. In der Boot-Konfiguration ist dann zusätzlich die Option `dokeymap` an die Kernel-Optionen anzuhängen.

Es gibt darüber hinaus einen Satz weiterer Optionen, die allerdings für den hier beschriebenen „Hausgebrauch“ kaum relevant sind.

2.2 Kernel-Erweiterungen

Es sollte nun ein Kernel zur Verfügung stehen, der zumindest eine Netzwerkkarte Ihres Systems unterstützt. Ist das nicht der Fall, hilft vermutlich nur ein externer Treiber weiter. In diesem Zusammenhang beschreiben wir die Integration externer Module in den Kernel, verlassen damit allerdings den Bereich, der auch für Anfänger als sicher gelten kann. Bei einer Erstinstallation sollten Sie, wenn irgend möglich, diesen Abschnitt überspringen und zu einem späteren Zeitpunkt als Referenz nutzen, um in einem stabil laufenden System eine noch nicht funktionierende Hardware-Komponente zu aktivieren.

Nicht alle Hardwaretreiber sind notwendigerweise schon im Kernel enthalten. Die Anforderungen an Code, der in den Linux-Kernel aufgenommen werden soll, sind hoch, und so gibt es zahlreiche Erweiterungen, die zwar (noch) nicht in den Kernel integriert sind (und es vielleicht auch nie sein werden), aber von den Entwicklern als externe Module angeboten werden. Falls ein solches Modul für die Unterstützung „exotischer“ Hardware notwendig ist, gibt es in den meisten Fällen auch das zugehörige Gentoo-Paket.

2.2.1 Externe Module automatisch laden

Nach der Installation eines Pakets, das ein externes Kernel-Modul bereitstellt, muss dieses auch beim Booten mit `modprobe` geladen werden. Vielfach ist der Gerätemanager `udev` (siehe auch Kapitel 16.4) in der Lage, ein Gerät und den notwendigen Treiber automatisch zu identifizieren und den entsprechenden `modprobe`-Befehl aufzurufen. Ist das allerdings nicht der

Fall und das neu installierte Modul wurde beim nächsten Booten oder dem Einstecken der Hardware nicht automatisch geladen, besteht die Möglichkeit, dies für den Boot-Vorgang explizit anzuweisen.

Nehmen wir an, wir benötigen zur Unterstützung einer Funknetzwerkkarte den Windows-Treiber und müssen dazu das Paket `net-wireless/ndiswrapper` installieren. Dieses installiert unter `/lib/modules/2.6.19-gentoo-r5` ein neues Kernel-Modul:

```
gentoo ~ # emerge -av net-wireless/ndiswrapper

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] sys-apps/pciutils-2.2.3-r2  0 kB
[ebuild  N    ] net-wireless/wireless-tools-28 USE="nls -multicall" 0 kB
[ebuild  N    ] net-wireless/ndiswrapper-1.33 USE="-debug -usb" 186 kB

Total: 3 packages (3 new), Size of downloads: 186 kB

Would you like to merge these packages? [Yes/No] Yes
...
>>> /lib/modules/2.6.19-gentoo-r5/misc/ndiswrapper.ko
```

Hinweis für Systeme mit Netzwerkanbindung

Die Installation von `net-wireless/ndiswrapper` setzt leider eine Internetverbindung voraus. Für eine mögliche Alternative siehe Seite 89.

Um das Modul beim Booten automatisch zu laden, ergänzen wir dessen Namen in der Datei `/etc/modules.autoload.d/kernel-2.6`, indem wir `ndiswrapper` über den `echo`-Befehl an die Liste anhängen:

```
gentoo ~ # echo "ndiswrapper" >> /etc/modules.autoload.d/kernel-2.6
gentoo ~ # cat /etc/modules.autoload.d/kernel-2.6
# /etc/modules.autoload.d/kernel-2.6:  kernel modules to load when syste
m boots.
#
# Note that this file is for 2.6 kernels.
#
# Add the names of modules that you'd like to load when the system
# starts into this file, one per line.  Comments begin with # and
# are ignored.  Read man modules.autoload for additional details.

# For example:
# aic7xxx
ndiswrapper
```

Das Skript `/etc/init.d/modules` ist dann dafür zuständig, die gelisteten Treiber während des Boot-Vorgangs zu laden.

2.2.2 Gerätenamen mit Treibern verknüpfen

Der Gerätemanager `udev` lädt Kernel-Module bisweilen auch aufgrund der Gerätenamen nach. So wird z. B. `wlan0` (bzw. `/dev/wlan0`) nach der Installation von `ndiswrapper` automatisch mit diesem Treiber identifiziert – sofern `udev` keinen anderen Treiber findet, der dieses Gerät noch spezifischer unterstützen könnte. Wir wollen uns hier aber nicht mit den Untiefen der `udev`-Konfiguration beschäftigen (vgl. dazu Kapitel 16.4), sondern uns ansehen, wie wir Gerätebezeichnungen mit dem Treiber-Namen assoziieren.

`ndiswrapper` installiert dafür eine kleine Datei in `/etc/modules.d`, ebenfalls mit dem Namen `ndiswrapper`.

```
gentoo ~ # cat /etc/modules.d/ndiswrapper
# modules.d configuration file for NDISWRAPPER

# Internal Aliases - Do not edit
# ------------------------------
alias wlan0 ndiswrapper


# Configurable module parameters
# ------------------------------
# if_name:Network interface name or template (default: wlan%d)
# proc_uid:The uid of the files created in /proc (default: 0).
# proc_gid:The gid of the files created in /proc (default: 0).
# debug:debug level
# hangcheck_interval:The interval, in seconds, for checking if driver is
 hung. (default: 0)
```

Mit der Zeile `alias wlan0 ndiswrapper` führt der Aufruf `modprobe wlan0` dazu, dass das `ndiswrapper`-Modul in den Kernel geladen wird.

Da ältere Versionen von `modprobe` nicht automatisch mit dem Verzeichnis `/etc/modules.d` umgehen können, sondern die Alias-Deklarationen in `/etc/modprobe.conf` erwarten, gibt es das Tool `modules-update`. Dieses fasst die Dateien in `/etc/modules.d` zu einer Datei `/etc/modprobe.conf` zusammen.

Nimmt man selbst Veränderungen an den Dateien in `/etc/modules.d` vor, um z. B. weitere Gerätenamen mit `ndiswrapper` zu verbinden, muss man anschließend `modules-update` aufrufen. Nach der Installation eines Paketes, das neue Dateien zu `/etc/modules.d` hinzufügt, ist dies nicht notwendig, da `emerge` das Werkzeug während der Installation automatisch aufruft.

Kapitel 3

Die Netzwerkkonfiguration

Mit Gentoo dauerhaft ohne Netzwerkverbindung arbeiten zu wollen ist natürlich nicht empfehlenswert. Das liegt vor allem daran, dass für die Installation neuer Pakete auch die Quellen benötigt werden, und auf der LiveDVD ist nur eine begrenzte Auswahl enthalten. Insgesamt umfassen die Quellen für alle in der Gentoo-Distribution enthaltenen Pakete ca. 50 GB.

Darüber hinaus verändert sich der Portage-Baum mit seinen Paket-Definitionen kontinuierlich, und Updates gibt es im Abstand von Minuten. Wie bei dieser Frequenz eine vernünftige Strategie zur Aktualisierung aussieht, erläutern wir in Kapitel 10 ab Seite 205. Ohne Netzwerkverbindung können wir jedoch in keinem Fall von neueren Versionen profitieren.

Also sollten wir an dieser Stelle zusehen, dass wir unseren frisch installierten Rechner mit einer Verbindung ins Internet ausstatten. Das Betriebssystem bemüht sich, wie bereits erwähnt, schon selbständig darum. Wer also schon zu diesem Zeitpunkt problemlos auf Seiten im Internet zugreifen kann sollte zum nächsten Kapitel springen und kann hierher zurückkehren, falls eine komplexere Konfiguration des Netzwerks ansteht.

Für ein funktionierendes Netzwerk sind drei Komponenten korrekt zu konfigurieren:

- der Kernel
- die Init-Skripte in `/etc/init.d`
- die eigentliche Konfigurationsdatei `/etc/conf.d/net`

Mit dem Kernel haben wir uns schon im vorigen Kapitel auseinander gesetzt und gehen davon aus, dass er so konfiguriert ist, dass zumindest eine Netzwerkschnittstelle unterstützt und korrekt angesprochen wird. Der Befehl `ifconfig -a` sollte also mindestens eine andere Schnittstelle als `lo` anzeigen (siehe auch Seite 25). Die Option `-a` bringt `ifconfig` dazu, auch Schnittstellen anzuzeigen, die noch nicht konfiguriert wurden.

Aber kümmern wir uns hier zunächst einmal um den zweiten Punkt: die Skripte in `/etc/init.d`, mit deren Hilfe wir eine Netzwerkschnittstelle starten bzw. stoppen.

3.1 Das Init-Skript einer Netzwerkschnittstelle

Unter 1.13.2 auf Seite 52 haben wir erwähnt, dass `/etc/init.d/net.lo` das zentrale Init-Skript ist, auf das letztlich jede Netzwerkschnittstelle zurückgreift. An der Benennung lässt sich schon erkennen, dass `net.lo` für die Initialisierung des Loopback-Interface zuständig ist. Weitere Schnittstellen können wir nach dem Schema `/etc/init.d/net.`*`interface`* hinzufügen. Jedes neue Skript dieser Art ist eine symbolische Verknüpfung auf `net.lo`.

Nehmen wir an, es gibt eine zusätzliche Schnittstelle `eth1` im System, so würden wir das zugehörige Skript folgendermaßen erstellen:

```
gentoo ~ # cd /etc/init.d
gentoo init.d # ln -s net.lo net.eth1
gentoo init.d # ls -la net.*
lrwxrwxrwx 1 root root   6 22. Jan 11:08 /etc/init.d/net.eth0 -> net.lo
lrwxrwxrwx 1 root root  18 25. Jan 10:28 /etc/init.d/net.eth1 -> /etc/i
nit.d/net.lo
-rwxr-xr-x 1 root root 30522  6. Apr 2007  /etc/init.d/net.lo
gentoo init.d # cd ~
gentoo ~ #
```

Mehr ist für das Init-Skript nicht zu tun. Es lässt sich nun mit den Kommandos `start`, `stop`, `restart` etc. verwenden. Näheres dazu findet sich im Kapitel 7 ab Seite 165.

Wollen wir die neue Schnittstelle beim Booten aktivieren, müssen wir sie zur Standard-Startsequenz hinzufügen (siehe Seite 52 und 165):

```
gentoo ~ # rc-update add net.eth1 default
 * net.eth1 added to runlevel default
```

Die eigentliche Konfiguration für die unter `init.d` angelegten Schnittstellen finden sich dann in den Konfigurationsdateien `/etc/conf.d/net` und `/etc/conf.d/wireless`.

3.2 Die automatisierte Konfiguration: net-setup

Schon auf der LiveDVD gibt es ein kleines Werkzeug namens `net-setup`, mit dem sich eine sehr einfache Netzwerkkonfiguration automatisiert erstellen lässt. Sein Funktionsumfang ist sehr begrenzt, aber wer davon ausgeht, dass sich der neue Rechner recht einfach in die Netzwerkumgebung einfügen müsste, und keine Lust hat, die Konfigurationsdateien zu editieren, der kann das Paket `app-misc/livecd-tools` nun installieren:

```
gentoo ~ # emerge -av app-misc/livecd-tools

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] sys-apps/pciutils-2.2.3-r2  0 kB
[ebuild  N    ] dev-util/dialog-1.0.20050206  USE="unicode" 0 kB
[ebuild  N    ] app-misc/livecd-tools-1.0.35-r1  USE="-X -opengl" 0 kB

Total: 3 packages (3 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
```

Damit lässt sich das Programm über `net-setup` aufrufen. Wir landen in einem rudimentären grafischen System, über das wir erkannte Netzwerkschnittstellen auswählen und konfigurieren. Wir entscheiden jeweils, ob diese kabel- oder WLAN-basiert ist und ob sie eine IP-Adresse im Netzwerk per DHCP bezieht.

Je nach Auswahl müssen mehr oder weniger Parameter definiert werden, und `net-setup` erstellt auf dieser Grundlage im Anschluss automatisch die Datei `/etc/conf.d/net`, so dass wir uns nicht um das korrekte Format kümmern müssen.

Wir wollen aber natürlich etwas tiefer in die Netzwerkkonfiguration einsteigen und es ermöglichen, den Rechner auch in einer eher ungewöhnlichen Netzwerkumgebung ans Netz zu bekommen.

3.3 /etc/conf.d/net

Die Konfiguration für jedwede Netzwerkschnittstelle eines Gentoo-Systems befindet sich in `/etc/conf.d/net`. Im Ursprungszustand enthält diese Datei die folgenden Zeilen:

```
# This blank configuration will automatically use DHCP for any net.*
# scripts in /etc/init.d.  To create a more complete configuration,
# please review /etc/conf.d/net.example and save your configuration
# in /etc/conf.d/net (this file :]!).
```

Gentoo versucht beim Aktivieren einer Netzwerkschnittstelle zunächst alle benötigten Informationen per DHCP (*Dynamic Host Configuration Protocol*) zu beziehen, wenn `/etc/conf.d/net` keine anderen Anweisungen enthält. Im einfachsten Fall ist damit eine Schnittstelle ordnungsgemäß zu konfigurieren und in Betrieb zu nehmen.

Wir wollen hier nur die häufigsten Varianten der Netzwerkkonfiguration beschreiben, da die Szenarien in diesem Bereich vor allem unter Linux sehr vielfältig sind. Für spezifischere Fragen bietet sich die Beispiel-Datei `/etc/conf.d/net.example` an, die einige Konfigurationsoptionen auflistet und knapp erklärt.

3.3.1 Grundlegendes

Die einzelnen Schnittstellen konfigurieren wir primär über die Variablen des Typs `config_`*`Schnittstelle`*. Dieser kann sehr unterschiedliche Werte enthalten, und die Netzwerkschnittstellen lassen sich so entsprechend der Hardware korrekt initialisieren. Die folgenden Zeilen sind zum Beispiel gültige Einträge für `/etc/conf.d/net`.

```
config_eth0=( "dhcp" )
config_eth1=( "192.168.178.2 netmask 255.255.255.0" )
config_eth2=( "noop" "192.168.0.2/24" )
```

Ein weiterer, allgemein gültiger Mechanismus für die Beeinflussung der Netzwerkkonfiguration ist die Auswahl von Netzwerk-Modulen. Dies kann generell für alle Schnittstellen über die Variable `modules` geschehen oder spezifisch für eine Schnittstelle in der gleichen Art und Weise wie oben über `modules_`*`Schnittstelle`*.

```
modules=( "dhcpclient" "!iwconfig" )
modules_eth0=( "dhcpcd" )
```

Alle weiteren notwendigen Parameter hängen dann von der spezifischen Konfigurationsvariante und den gewählten Modulen ab. Die gebräuchlichsten Varianten wollen wir im Folgenden beleuchten.

3.4 DHCP

Den Standardfall der Netzwerkkonfiguration stellt das *Dynamic Host Configuration Protocol* (DHCP) dar. Klinken wir unseren Rechner in ein Netzwerk ein, in dem es einen DHCP-Server gibt, so ist das System in der Lage, über eine anfangs unkonfigurierte Netzwerkschnittstelle eine allgemeine Anfrage nach den korrekten Konfigurationswerten in das Netz zu senden. Der DHCP-Server ist im nächsten Schritt dafür zuständig, die notwendigen Daten, wie z. B. die IP-Adresse, an unseren Rechner zurück zu schicken. Diese Angaben dienen unserem System dann zur vollautomatischen Konfiguration der Netzwerkverbindung.

Wie bereits erwähnt, ist der Standardfall – nämlich gar keine Angabe zur Konfiguration einer Schnittstelle in `/etc/conf.d/net` – über DHCP abgedeckt. Wir können die Verwendung von DHCP aber auch explizit für eine Schnittstelle definieren bzw. anfordern:

```
config_eth0=( "dhcp" )
```

Ganz von alleine spricht Linux allerdings kein DHCP. Dafür ist ein spezifisches Programm notwendig, das mit dem DHCP-Protokoll umgehen kann. Standardmäßig haben wir in Kapitel 1.13.1 ab Seite 51 das Paket `net-misc/dhcpcd` als DHCP-Client installiert, das die Netzwerkskripte im Normalfall nutzen. Es gibt allerdings noch drei alternative Clients, und zwar `dhclient` (Paket `net-misc/dhcp`), `pump` (Paket `net-misc/pump`) und `udhcpc` (Paket `net-misc/udhcp`).

Möchte man einem dieser drei Pakete den Vorzug geben, muss man es über `emerge` installieren und gleichzeitig das zugehörige Netzwerk-Modul in `/etc/conf.d/net` aktivieren. Folgendes würde `dhclient` auswählen:

```
modules=( "dhclient" )
```

Wir werden dieser Art, Module in `/etc/conf.d/net` zu aktivieren, noch einige Male begegnen. Wie bereits erwähnt, kann man diese Module auch schnittstellenspezifisch zuweisen:

```
modules_eth0=( "dhcpcd" )
modules_eth1=( "dhclient" )
```

Jeder Client besitzt eigene Optionen, die sich ebenfalls in der Netzwerk-Konfigurationsdatei festlegen lassen. Häufig definiert wird z. B. der Parameter für den Timeout, bis zu dem der Client eine Antwort des DHCP-Servers erwartet. Bei `dhcpcd` legen wir diese Zeitspanne mit der Option `-t` fest und spezifizieren dies in `/etc/conf.d/net` folgendermaßen:

```
dhcpcd_eth0="-t 10"
```

Damit wird als Option für das `dhcpcd`-Modul beim Initialisieren der `eth0`-Schnittstelle der Timeout-Wert auf 10 Sekunden gesetzt. Erhält der Client nach 10 Sekunden keine Antwort vom Server, betrachtet er die automatische Konfiguration als fehlgeschlagen. Standardmäßig liegt dieser Wert bei 20 Sekunden, was den Boot-Vorgang verzögern kann, wenn kein Netzwerkkabel angeschlossen oder kein DHCP-Server zur Verfügung steht.

3.4.1 DHCP-Timeout

Fehlt das Netzwerkkabel, hilft eine DHCP-Anfrage nicht weiter. Der DHCP-Client wird sein Signal vergeblich senden und mit Sicherheit keine Antwort erhalten. Sofern die Netzwerkkarte überhaupt nicht mit einem Kabel an einen Netzwerk-Hub angeschlossen ist, sollte man auf die entsprechende Anfrage ganz verzichten. Hier kommt das ab Seite 83 besprochene `netplug` ins Spiel.

Ist ein Netzwerkkabel angeschlossen, aber kein DHCP-Server verfügbar, lässt sich zwar die DHCP-Anfrage senden, aber auf Antwort wartet der Rechner ebenso vergeblich. Befindet sich unsere Maschine konstant im Netzwerk, vergeben wir die IP-Adressen am besten statisch nach der ab Seite 84 beschriebenen Methode.

Laptops befinden sich gelegentlich in Netzen mit DHCP-Server, dann wieder in einer Umgebung, in der es dieses Hilfsmittel nicht gibt. Die notwendige Konfiguration beschreiben wir im Folgenden.

Fehlender DHCP-Server

Auch in einem Netz ohne DHCP-Server ist es sinnvoll, eine IP-Adresse zu vergeben und den Rechner damit ansprechbar zu machen. Gerade mobile Geräte sollten also für den Bedarfsfall eine statische Adresse erhalten. Das lässt sich mit Hilfe einer *Fallback*-Adresse erreichen:

```
fallback_eth0=( "192.168.178.2 netmask 255.255.255.0" )
fallback_route_eth0=( "default via 192.168.178.1" )
```

Das Format für diese Fallback-Adresse ist dasselbe wie für die Vergabe statischer Adressen, und wir schauen es uns in Kapitel 3.5 nochmals genauer an.

Gibt es im Netz keinen DHCP-Server, versagt die automatische Konfiguration und die Netzwerkschnittstelle würde im Normallfall nicht initialisiert. Ist die Fallback-Adresse gesetzt, setzen die Netzwerkskripte nach dem Timeout die festgelegte statische Konfiguration.

3.4.2 netplug

Fehlt die Verbindung zum Netzwerk schon auf physikalischer Ebene, ist der Versuch, die Netzwerkschnittstelle zu konfigurieren bzw. DHCP-Anfragen in das Netz zu schicken, sinnlos. Bleibt die Frage, ob wir das Fehlen des Kabels detektieren können.

Ethernet-Karten bauen auch im Ruhezustand ein Carrier-Signal zum Hub auf, sobald ein Kabel die beiden verbindet. Dieses Signal kann der Kernel erkennen, sofern die Treiber der Netzwerkkarte dies unterstützen. Das Paket `sys-apps/netplug` liefert den `netplugd`-Service, der die verfügbaren Netzwerkschnittstellen konstant überwacht und hochfährt, sobald der Kernel das Carrier-Signal erkennt. Eine Alternative zu `sys-apps/netplug` ist `sys-apps/ifplugd`.

Wir installieren hier `sys-apps/netplug`:

```
gentoo ~ # emerge -av sys-apps/netplug

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] sys-apps/netplug-1.2.9-r3  0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
```

Im Grunde brauchen wir keine weitere Konfiguration, denn die Netzwerk-Skripte erkennen das Paket und verwenden es automatisch. `netplugd` funktioniert denkbar einfach: Sobald das Programm auf einer der überwachten Schnittstellen eine Veränderung des Carrier-Signals erkennt, startet bzw. stoppt es diese über das `/etc/init.d`-Skript. Sämtliche Konfigurationen in `/etc/conf.d/net` sind weiterhin gültig.

Wer `netplugd` nicht einsetzen möchte, obwohl das Paket installiert ist, schließt das Modul in `/etc/conf.d/net` aus:

```
modules=( "!netplug" )
```

Mit dieser Syntax lässt sich das Modul auch für einzelne Schnittstellen deaktivieren:

```
modules_eth0=( "!netplug" )
```

Auch `ifplugd` verwendet diese Syntax; mit der allgemeinen Bezeichnung `plug` spricht man beide Module zugleich an:

```
modules=( "!plug" )
```

Hier sind die Plug-Services für alle Netzwerkschnittstellen unterbunden. Sie können `netplugd` auch nur für bestimmte Netzwerkschnittstellen verwenden, indem Sie `/etc/netplug/netplugd.conf` modifizieren:

```
gentoo ~ # cat /etc/netplug/netplugd.conf
eth*
```

Soll der Service nur für einzelne Schnittstellen aktiviert sein, listet man diese auf:

```
gentoo ~ # cat /etc/netplug/netplugd.conf
eth0
eth2
```

3.5 Statische IP

Wer im Netzwerk keinen DHCP-Server betreibt, kann einzelnen Rechnern Adressen auch statisch zuweisen; neben der IP-Adresse ist die Angabe der *Netmask* notwendig. Diese zeigt an, für welche Bereiche des Netzwerks kein Routing notwendig ist.

Nehmen wir an, wir wollen unserem Rechner die Adresse `192.168.178.2` geben und alle Rechner `192.168.178.*` sind Teil des lokalen Netzwerks, so lautet der Eintrag:

```
config_eth0=( "192.168.178.2 netmask 255.255.255.0" )
```

Verkürzt lässt sich die IP-Adresse in Kombination mit der Netzwerkmaske[1] auch so darstellen:

```
config_eth0=( "192.168.178.2/24" )
```

Für IP-Adressen, die nicht mit `192.168.178` starten, wird der Rechner dann versuchen, einen Router anzusprechen. In der obigen Konfiguration fehlt aber noch die Angabe, wie dieser Router zu erreichen ist. Das erreichen wir mit einem Eintrag `routes_`*`Schnittstelle`*.

```
routes_eth0=( "default via 192.168.178.1" )
```

Hier ist der Standard-Gateway in andere Bereiche des Internets der Rechner mit der IP `192.168.178.1`.

Es spricht nichts dagegen, weitere Einträge für die Routing-Tabelle anzugeben.

[1] `http://de.wikipedia.org/wiki/Subnetz`

```
routes_eth0=( "default via 192.168.178.1"
              "10.0.0.0/8 via 192.168.178.100" )
```

Hier sprechen wir das Subnetz `10.*.*.*` über den Rechner mit der IP `192.168.178.100` an. Alle anderen Verbindungen in die Außenwelt laufen über `192.168.178.1`.

3.6 Modem

Zur Unterstützung eines analogen Modems muss das Paket `net-dialup/ppp` installiert sein. Dieses liefert das Verbindungstool `pppd`, das die Kommunikation mit der Gegenstelle abwickelt.

```
gentoo ~ # emerge -av net-dialup/ppp

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-dialup/ppp-2.4.4-r4  USE="ipv6 pam -activefilter
-atm -dhcp -eap-tls -gtk -mppe-mppc -radius" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
```

Außerdem sollte man überprüfen, ob der Kernel das *Point-to-Point*-Protokoll (PPP) aktiviert hat; sehen wir uns dazu im Verzeichnis `/usr/src/linux` mit `make menuconfig` die Kerneloptionen an, und zwar unter **Device Drivers | Network Device Support | PPP (point-to-point protocol) support**.

Müssen wir die Konfiguration ändern, greifen wir wieder auf `genkernel` zurück und kompilieren den Kernel neu:

```
gentoo ~ # genkernel --menuconfig --kernname=ppp --no-clean all
```

Anschließend können wir das Modul `ppp` einer Netzwerkschnittstelle zuweisen, beispielsweise `ppp0`. In `/etc/conf.d/net` fügen wir folgende Zeile ein, damit die Kommunikation über diese Schnittstelle per PPP erfolgt:

```
config_ppp0=( "ppp" )
```

Gleichzeitig benötigen wir das Init-Skript für die Schnittstelle und erstellen den entsprechenden Link unter `/etc/init.d/`.

```
gentoo ~ # ln -s /etc/init.d/net.lo /etc/init.d/net.ppp0
```

Da die meisten Modemzugänge in das Internet zeitbasiert abrechnen, wollen wir die Schnittstelle vermutlich nur bei Bedarf aktivieren und sie nicht über `rc-update` zum Boot-Vorgang hinzufügen.

Nun fehlt noch die eigentliche Konfiguration für die Schnittstelle, insbesondere der Gerätename für das Modem. `pppd` benötigt diesen für die Kommunikation mit der Gegenstelle. Er lautet meist `/dev/ttyS0` (für eine serielle Schnittstelle), und wir legen ihn über den Parameter `link_`*`Schnittstelle`* fest.

```
link_ppp0=( "/dev/ttyS0" )
```

Darüber hinaus benötigen wir die Telefonnummer des Providers sowie Benutzernamen und Passwort. Nehmen wir an, wir wählen uns bei Arcor ein:

```
phone_number_ppp0=( "01920787" )
username_ppp0="arcor"
password_ppp0="internet"
```

Der `pppd`-Daemon akzeptiert zahlreiche Optionen, die über die Variable `pppd_`*`Schnittstelle`* in der Datei `/etc/conf.d/net` für jede Modem-Verbindung separat definiert werden können.

```
pppd_ppp0=(
        "defaultroute"
        "usepeerdns"
        "debug"
        "noauth"
)
```

`defaultroute` bestimmt die Modem-Verbindung als Standard-Gateway zur Außenwelt.

`usepeerdns` sorgt dafür, dass `pppd` die DNS-Parameter vom Provider abfragt und in die Datei `/etc/resolv.conf` schreibt. Die meisten Provider liefern die DNS-Konfiguration automatisiert, und darum sollte die Option üblicherweise aktiviert sein.

`debug` kann vor allem beim ersten Testen der Verbindung sinnvoll sein, um bei Problemen einen Hinweis auf die Ursache im System-Log zu erhalten.

`noauth` verhindert, dass `pppd` vom Provider eine Authentifizierung verlangt, die ein allgemeiner Internetprovider auch nicht liefert.

Zu guter Letzt fehlt noch ein so genanntes *Chat-Skript*. Dieses kümmert sich um den Ablauf der Kommunikation über die `pppd`-Verbindung und sieht folgendermaßen aus:

```
chat_ppp0=(
        'ABORT' 'BUSY'
        'ABORT' 'ERROR'
        'ABORT' 'NO ANSWER'
        'ABORT' 'NO CARRIER'
        'ABORT' 'NO DIALTONE'
        'ABORT' 'Invalid Login'
        'ABORT' 'Login incorrect'
        'TIMEOUT' '5'
        '' 'ATZ'
        'OK' 'ATDT\T'
        'TIMEOUT' '60'
        'CONNECT' ''
        'TIMEOUT' '5'
        '~--' ''
)
```

3.7 WLAN

Den WLAN-Zugang unter Linux einzurichten war lange Zeit ein Abenteuer, weil Treiber gar nicht oder nur im experimentellen Stadium zur Verfügung standen. Mittlerweile hat sich die Situation verbessert, aber manche Geräte treiben Nutzer immer noch zur Verzweiflung. Sofern die Möglichkeit besteht, sollte man sich vor dem Kauf der Hardware nach verfügbaren Treibern umsehen. Denn setzt man Hardware ein, die sich problemlos unter Linux ansprechen lässt, ist auch die Konfiguration meist leicht zu bewältigen. Hier wollen wir zunächst noch einmal kurz auf die Kernelkonfiguration zurückkommen.

3.7.1 WLAN-Treiber

Im einfachsten Fall ist ein Treiber in der **Wireless**-Sektion der Kernel-Treiber verfügbar. Wie in 3.6 beschrieben, schauen wir uns die entsprechende Sektion in der Kernel-Konfiguration an: **Device Drivers | Network Device Support | Wireless LAN (non-hamradio)**.

Hier muss in jedem Fall die Option **Wireless LAN drivers** aktiviert sein, andernfalls ist die Treiberauswahl nicht verfügbar. Mit den hier aufgelisteten Treibern hat man die höchsten Chancen, die eigene Hardware nervenschonend zum Laufen zu bringen.

Passt jedoch keiner der gelisteten Treiber auf das eigene Modell, bleibt noch die Auswahl der Kernelmodule der Kategorie `net-wireless`. Wir suchen hier einmal mit `grep` nach der Markierung `linux-mod` in den Ebuilds. Damit identifizieren wir Pakete, die einen Kernel-Treiber bereitstellen. Aus der

resultierenden Liste schneiden wir den Paketnamen mit `cut` heraus und bemühen `uniq`, damit jeder Name wirklich nur einmal gelistet wird:

```
gentoo ~ # grep linux-mod /usr/portage/net-wireless/**/*.ebuild | \
> cut -f 5 -d "/" | uniq
acx
adm8211
at76c503a
fwlanusb
hostap-driver
ieee80211
ipw2100
ipw2200
ipw3945
linux-wlan-ng
linux-wlan-ng-modules
madwifi-ng
madwifi-old
mcs7780
ndiswrapper
orinoco
prism54
ralink-rt61
rfswitch
rt2400
rt2500
rt2570
rt2x00
rt61
rtl8180
rtl8187
zd1201
zd1211
```

Diese Treiber weisen als externe Kernel-Module in den meisten Fällen nicht die gleiche Stabilität wie die integrierten Treiber auf, und folglich ist hier eher mit Problemen zu rechnen.

Verwendet man eines dieser Module, darf man nicht vergessen, in der oben angegebenen Kernel-Sektion die allgemeine Option **Wireless LAN drivers** zu aktivieren. Andernfalls unterstützt der Kernel kein WLAN und das externe Modul kann nicht korrekt arbeiten.

Eine Sonderstellung nimmt das Paket `net-wireless/ndiswrapper` ein. Es ist soz. der letzte Rettungsanker für die WLAN-Hardware-Unterstützung, wenn alle anderen Maßnahmen fehlschlagen. Dieses Modul erlaubt die Nutzung der für alle WLAN-Karten natürlich verfügbaren Windows-Treiber innerhalb des Linux-Kernels. Das klingt abenteuerlicher als es ist, und dieses Verfahren der Hardware-Ansteuerung führt recht häufig zum Erfolg. Wir haben uns bereits im vorigen Kapitel auf Seite 75 mit dem Paket beschäf-

tigt, als es um zusätzliche Kernel-Module ging. Hier wollen wir uns mit der Funktionalität des Paketes beschäftigen.

Hinweis für Systeme mit Netzwerkanbindung

Leider ist der Quellcode zu `net-wireless/ndiswrapper` nicht auf der LiveDVD vorhanden, so dass Sie eine schon bestehende Netzwerkverbindung für die Installation benötigen. Sollte die WLAN-Karte der einzige Weg sein, die Maschine ins Netz zu bekommen, sollten sie sich das Quellpaket[2] einzeln herunterladen und in `/usr/portage/distfiles` legen.

net-wireless/ndiswrapper

Der Windows-Treiber besteht aus einer `*.sys`- und einer `*.inf`-Datei, die wir gegebenenfalls noch entpacken und in einem Verzeichnis ablegen müssen. Anschließend stehen die Dateien `ndiswrapper` zur Verfügung. Wir haben sie im folgenden Beispiel in `/tmp/wlan` platziert:

```
gentoo ~ # la /tmp/wlan
-rw-r--r--  1 wrobel users   8398 2007-02-27 00:27 PRISMA00.inf
-rw-r--r--  1 wrobel users 380736 2007-02-27 00:27 PRISMA00.sys
gentoo ~ # ndiswrapper -i /tmp/wlan/PRISMA00.inf
```

Mit der Option `-i` installieren wir einen Windows-Treiber, der damit zur Verfügung steht, so dass `ndiswrapper` die Karte beim Startvorgang erkennt. Damit das Modul automatisch geladen wird, fügen wir es zu der Datei `/etc/modules.autoload.d/kernel-2.6` hinzu (bzw. `kernel-2.4`, wenn man einen älteren Kernel verwendet).

```
gentoo ~ # echo "ndiswrapper" >> /etc/modules.autoload.d/kernel-2.6
```

3.7.2 WLAN-Konfiguration: iwconfig oder wpa_supplicant

Es gibt zwei Wege der WLAN-Konfiguration: über `iwconfig` oder mit Hilfe von `wpa_supplicant`. Mit letzterem lassen sich auch WPA-verschlüsselte Netzwerke nutzen, und es ist aufgrund der Sicherheitsproblematik bei Funknetzwerken vorzuziehen.

Die Konfiguration über `iwconfig` erfolgt in der bereits bekannten Datei `/etc/conf.d/net`, während `wpa_supplicant` davon leider abweicht. Wir wollen hier beide Alternativen beschreiben.

[2] `ndiswrapper-1.33.tar.gz` von `http://ndiswrapper.sourceforge.net/`

iwconfig

Das Programm `iwconfig` installieren wir über das Paket `net-wireless/wireless-tools`. Wer sich mit einem unverschlüsselten oder über WEP (*Wired Equivalent Privacy*) verschlüsselten Netzwerk verbinden möchte, installiert das Paket hier:

```
gentoo ~ # emerge -av net-wireless/wireless-tools

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-wireless/wireless-tools-28  USE="nls -multicall" 0 k
B

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
```

`iwconfig` sollte in der Lage sein, vorhandene WLAN-Schnittstellen anzuzeigen:

```
gentoo ~ # iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11b/g  ESSID:""
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Encryption key:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Die Karte `wlan0` ist noch nicht konfiguriert, und wir bearbeiten dazu wieder `/etc/conf.d/net`. In den meisten Fällen ist die Konfiguration recht simpel, und wir werden hier auch nicht auf Sonderfälle eingehen, von denen viele in der Datei `/etc/conf.d/wireless.example` ausführlich diskutiert werden.

Im einfachsten Fall genügt es, für die Netzwerkschnittstelle den Einsatz von `iwconfig` und für einen bestimmten WLAN-Netzwerk-Namen die SSID (*Service Set Identifier*) sowie einen Schlüssel festzulegen:

```
modules=( "iwconfig" )
key_GENTOO=( "s:meingeheimerschlüssel" )
```

Wie bereits ausgeführt, lassen sich diese Modulangaben bei Bedarf auch schnittstellenspezifisch treffen.

In der zweiten Zeile definieren wir für das WLAN-Netzwerk mit der SSID `GENTOO` den Schlüssel. Dieser ist hier als String angegeben (daher das führende `s:`). Wir könnten den Schlüssel auch als Zahlenfolge in der Form `1234-1234-1234-1234-1234-1234-56` angeben.

Bei kabelgebundenen Netzwerken handelt es sich vielfach um längerfristige Verbindungen. Die notwendigen Parameter (IP-Adresse, DHCP etc.) sind oft abhängig von der jeweiligen Schnittstelle, weshalb wir den Parameter mit `config_eth0` spezifizieren.

Bei einer WLAN-Schnittstelle können das Netzwerk und die dafür notwendigen Parameter deutlich schneller wechseln, und es ist darum wenig sinnvoll, sich auf den Namen der Netzwerkschnittstelle zu beziehen. Der Name des WLAN-Netzwerks verspricht da mehr Konstanz. Aus diesem Grunde lassen sich die bisher bekannten Variablen nicht nur mit dem Schnittstellennamen verknüpfen, sondern auch mit der SSID.

Um also festzulegen, dass wir im Netzwerk `GENTOO` DHCP einsetzen, schreiben wir:

```
config_GENTOO=( "dhcp" )
```

Analog ist das Vorgehen bei anderen Parametern, die wir bei der Konfiguration einer Netzwerkschnittstelle bereits kennen gelernt haben.

wpa_supplicant

Für WPA-verschlüsselte (*Wi-Fi Protected Access*) Netzwerke benötigen wir das Paket `net-wireless/wpa_supplicant` und installieren es mit

```
gentoo ~ # emerge -av net-wireless/wpa_supplicant

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-wireless/wpa_supplicant-0.5.7  USE="readline ssl
-dbus -gnutls -gsm -madwifi -qt3 -qt4" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
```

Nur ein kleiner Teil der notwendigen Konfiguration unseres Netzwerks gehört in die Datei `/etc/conf.d/net`:

```
modules=( "wpa_supplicant" )
wpa_supplicant_wlan0="-Dwext"
```

Mit der ersten Zeile bestimmen wir, dass der Zugriff auf ein WPA-gesichertes Netzwerk erfolgt. In der zweiten Zeile geben wir Optionen mit auf den Weg, von denen jene für den Treiber-Typ (`-D`) die wichtigste ist.

`wpa_supplicant` unterstützt nur eine begrenzte Zahl an Treiber-Typen. Die genaue Liste erhalten wir über die Hilfe des Programms:

```
gentoo ~ # wpa_supplicant -h
...

drivers:
  wext = Linux wireless extensions (generic)
  hostap = Host AP driver (Intersil Prism2/2.5/3)
  prism54 = Prism54.org driver (Intersil Prism GT/Duette/Indigo)
  atmel = ATMEL AT76C5XXx (USB, PCMCIA)
  ndiswrapper = Linux ndiswrapper
  ipw = Intel ipw2100/2200 driver (old; use wext with Linux 2.6.13 or ne
wer)
  wired = wpa_supplicant wired Ethernet driver

...
```

Hier ein Beispiel aus der Praxis, in dem wir einen USB-WLAN-Stick testen, der vom Treiber `net-wireless/fwlanusb` unterstützt wird. Dieser wiederum beherrscht die *Wireless Extensions* von Linux, so dass wir `wpa_supplicant` die Option `-Dwext` mit auf den Weg geben. Dass der WLAN-Stick diesen Treiber benötigt, haben wir zuvor der Dokumentation des Paketes `net-wireless/fwlanusb` entnommen.

Wenn der Kernel die WLAN-Karte aber nun eigenständig erkannt hat, so lässt sich die Frage, welcher Treiber denn die WLAN-Karte überhaupt bedient, nicht ganz so einfach beantworten. Hilfe bietet das Paket `sys-apps/ethtool`:

```
gentoo ~ # emerge -av sys-apps/ethtool

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] sys-apps/ethtool-4  0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
```

Wir wollen uns hier nur eine Option des Programms `ethtool` ansehen: `--driver` (bzw. `-i`). Diese liefert den Namen des Treibers zu unserer Netzwerkschnittstelle:

```
gentoo ~ # ethtool -i eth1
driver: prism54
```

```
version: 1.2
firmware-version:
bus-info:
```

Bei dieser Karte wäre also die Option `-Dprism54` für `wpa_supplicant` die richtige Wahl.

Zwar sind die Vorarbeiten in `/etc/conf.d/net` abgeschlossen, aber der Hauptteil der Konfiguration erfolgt in `/etc/wpa_supplicant/wpa_supplicant.conf`. Leider gibt es recht viele Konfigurationsvarianten, die wir hier nicht alle besprechen können. Eine dokumentierte Beispieldatei findet sich unter `/usr/share/doc/wpa_supplicant-0.5.7/wpa_supplicant.conf.bz2` (wenn die Version `net-wireless/wpa_supplicant-0.5.7` installiert ist).

Wir gehen hier nur beispielhaft auf eine mögliche Variante ein:

```
gentoo ~ # cat /etc/wpa_supplicant/wpa_supplicant.conf
network={
        ssid="GENTOO"
        key_mgmt=WPA-PSK
        proto=WPA2
        pairwise=CCMP
        group=CCMP
        psk="1234567890123456"
}
```

Das umschließende Statement `network={...}` zeigt an, dass es sich um den Zugang zu einem WLAN-System handelt. Wir können mehrere solcher Blocks definieren, und `wpa_supplicant` geht sie der Reihe nach durch, bis das Programm ein Netzwerk findet, zu dem sich eine Verbindung aufbauen lässt.

Den Netzwerknamen (die SSID) legen wir hier mit `ssid` fest. Die Angabe ist zwingend notwendig; in unserem Beispiel heißt das Netzwerk `GENTOO`.

Der `key_mgmt`-Eintrag legt die Schlüssel-Methode fest. Standardmäßig hat diese Option den Wert `WPA-PSK WPA-EAP` und akzeptiert damit sowohl Pre-Shared-Key-Verfahren (PSK) als auch erweiterte Authentifizierungsprotokolle (*Extensible Authentication Protocol*, EAP). In den meisten Fällen kommt der Pre-Shared-Key zum Einsatz, und wir beschränken die Verbindung hier mit `key_mgmt=WPA-PSK` auf dieses Verfahren. Wer `wpa_supplicant` für WEP-Verbindungen einsetzen möchte, muss `key_mgmt` explizit auf `NONE` setzen.

Im Beispiel bietet das WLAN-Netz WPA2-Verschlüsselung, und so akzeptieren wir dieses Protokoll in `proto`. WPA2 verschlüsselt über `CCMD`[3], und darum müssen wir diese Methode sowohl für `pairwise` (*Unicast*; Punkt-zu-

[3] `http://en.wikipedia.org/wiki/CCMP`

Punkt-IP-Verkehr) als auch `group` (*Broadcast, Multicast*; Punkt-zu-Gruppe-IP-Verkehr) angeben. Das ältere WPA setzt auf TKIP[4] als Verschlüsselungsverfahren, und die Einträge müssen für ein WPA-verschlüsseltes WLAN-Netzwerk wie folgt aussehen:

```
...
        key_mgmt=WPA-PSK
        proto=WPA
        pairwise=TKIP
        group=TKIP
...
```

Schließlich das Wichtigste: der Pre-Shared-Key in der Option `psk`.

Wir haben hier die meisten Werte explizit gesetzt, aber mit etwas Glück ist `wpa_supplicant` auch in der Lage, Parameter selbständig zu ermitteln. Wir definieren dann lediglich den Netznamen und den Schlüssel. Das geht allerdings nur, wenn wir über WPA(2) verschlüsseln und nicht WEP:

```
network={
        ssid="gentoo"
        psk="1234567890123456"
}
```

Damit bleibt nur zu hoffen, dass Ihr System spätestens jetzt erfolgreich mit dem Internet in Verbindung steht, so dass wir uns nun mit den Dingen beschäftigen, die Gentoo eigentlich ausmachen.

[4] `http://en.wikipedia.org/wiki/Temporal_Key_Integrity_Protocol`

Kapitel 4

Paketmanagement mit emerge

Nach der Installation und der Konfiguration des Netzwerkzugangs sind wir nun endlich soweit, dass wir uns mit den Elementen beschäftigen, die Gentoo letztlich zu dem machen, was es ist: der Paketverwaltung *Portage*.

Am Anfang der Entwicklung von Gentoo stand u. a. der Wunsch nach einer Linux-Distribution, deren Softwareverwaltung demselben Prinzip folgt wie das Ports-System von FreeBSD. Entsprechend heißt das Gentoo-Paketmanagementsystem *Portage*. Wie bei FreeBSD steuert eine zentrale Konfigurationsdatei namens `/etc/make.conf` die Arbeitsweise dieser Software und damit des Gentoo-Systems.

Wir haben uns in der Einleitung (Seite 15) schon kurz mit dem Namen *Portage* auseinander gesetzt, wollen aber hier aber noch einmal kurz die Nomenklatur aufgreifen, damit wir sicher sind, dass die Begrifflichkeiten der folgenden Kapitel präzise definiert sind. Konkret geht es uns dabei um das Verhältnis dreier Elemente: *Portage*, `emerge` und den *Portage-Baum*.

Portage bezeichnet das Paketmanagement-System von Gentoo im Ganzen, d. h. es ist ein Überbegriff, der sowohl die Werkzeuge für das Paketmanage-

ment beinhaltet als auch die zugrunde liegenden Programmbibliotheken und die Paketdefinitionen. Das Tool `emerge` ist unser primärer Zugang zu Portage. Es ist unser wichtigstes Werkzeug zum Paketmanagement. Es repräsentiert somit Portage und wir sprechen gelegentlich auch davon, dass Portage eine Aktion durchführt, wenn wir als Benutzer `emerge` aufgerufen haben. Der Portage-Baum bezeichnet die eigentlichen Paketdefinitionen unter `/usr/portage`. Ohne sie würde Portage zwar auch existieren, hätte allerdings keinerlei Pakete zu managen.

Wir wollen uns im Folgenden zuerst einmal mit der Funktionsweise von Portage auseinander setzen. Dabei werden wir vor allem lernen, wie wir neue Software in unserem derzeit noch sehr reduzierten System installieren. Erst in Kapitel 6 beschäftigen wir uns dann in aller Ausführlichkeit mit `/etc/make.conf`, der wichtigsten Konfigurationsdatei des Portage-Systems.

4.1 emerge

`emerge` ist das zentrale Tool eines Gentoo-Systems. Wir haben uns während der Installation nur kurz mit der Basis-Funktionalität beschäftigt und vor allem die Grundlagen der Paketbezeichnung erklärt (vgl. Kapitel 1.7.1 ab Seite 41). Nun tauchen wir tiefer in das Paketmanagement ein und lernen die Handhabung des Werkzeugs auf der Kommandozeile kennen.

Eine seiner wichtigsten Optionen für den Gentoo-Einsteiger sei vorweg erwähnt: `emerge --help`.

```
gentoo ~ # emerge --help
Usage:
   emerge [ options ] [ action ] [ ebuildfile | tbz2file | dependency ]
[ ... ]
   emerge [ options ] [ action ] < system | world >
   emerge < --sync | --metadata | --info >
   emerge --resume [ --pretend | --ask | --skipfirst ]
   emerge --help [ system | world | config | --sync ]
Options: -[abBcCdDefgGhikKlnNoOpqPsStuvV] [--oneshot] [--newuse]
[--noconfmem][ --color < y | n >  ][ --columns ][--nospinner][ --deep  ]
[--with-bdeps < y | n > ]
Actions: [ --clean | --depclean | --prune | --regen | --search | --unmer
ge ]
...
```

Die Option liefert, abgesehen von der oben dargestellten Übersicht, eine detaillierte Erklärung jeder Option. Neuere Versionen von `emerge` liefern mit `--help` nur diesen Überblick und erst mit der zusätzlichen Option `--verbose` die komplette Liste.

4.2 Grundfunktionen

Eine der zentralen Aufgaben eines Paketmanagement-Systems ist es, dafür zu sorgen, dass die Abhängigkeiten zwischen allen installierten Paketen erfüllt sind. Während wir uns im letzten Kapitel nur damit beschäftigt haben, einzelne Pakete zu installieren, gehen wir an dieser Stelle einen Schritt weiter und sehen uns an, wie `emerge` reagiert, wenn wir ein Paket installieren wollen, das weitere, nicht installierte Pakete benötigt. Als Beispiel wollen wir für unseren Webserver den Apache-Server installieren.

4.2.1 Die Installation simulieren

Den Server installieren wir mit dem Paket `net-www/apache`:

```
gentoo ~ # emerge -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-nds/openldap-2.3.30-r2  USE="berkdb crypt gdbm ipv6
perl readline ssl tcpd -debug -kerberos -minimal -odbc -overlays -samba
-sasl (-selinux) -slp -smbkrb5passwd" 0 kB
[ebuild  N    ] dev-libs/apr-0.9.12  USE="ipv6 -urandom" 0 kB
[ebuild  N    ] app-misc/mime-types-5  0 kB
[ebuild  N    ] dev-libs/apr-util-0.9.12  USE="berkdb gdbm ldap" 0 kB
[ebuild  N    ] net-www/apache-2.0.58-r2  USE="apache2 ldap ssl -debug -
doc -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mpm-
worker (-selinux) -static-modules -threads" 4,652 kB

Total: 5 packages (5 new), Size of downloads: 4,652 kB
```

In diesem Fall verwenden wir die Flags `-p` (bzw. `--pretend`) und `-v` (bzw. `--verbose`), um `emerge` mitzuteilen, dass wir nur sehen möchten, was installiert werden *würde*. Die Option `--verbose` bewirkt hier, dass auch die Größe des herunterzuladenden Pakets angezeigt wird.

Hinweis für Systeme mit Netzwerkanbindung

Haben wir das System schon mit `emerge --sync` über das Netzwerk aktualisiert, so findet sich der Apache-Server nicht mehr in der Kategorie `net-www` wieder, sondern unter `www-servers`.

4.2.2 Paketabhängigkeiten

Für eine erfolgreiche Installation des Apache sind also vier weitere, derzeit nicht installierte Pakete notwendig: `dev-libs/apr`, `dev-libs/apr-util`, `app-misc/mime-types` und `net-nds/openldap`. Die Pakete sind am Anfang der Zeile mit dem Buchstaben `N` für „new“ markiert (`[ebuild N ]`). Die Markierung steht für ein derzeit nicht im System installiertes Paket, das `emerge` neu hinzufügen müsste.

Dass hier der LDAP-Server `net-nds/openldap` installiert werden soll, liegt an dem `ldap`-USE-Flag, das wir in Kapitel 1.9 aktiviert haben. Wir werden später ab Seite 114 auf diesen Punkt eingehen.

Portage wird diese Pakete entsprechend der angegebenen Reihenfolge installieren, und so können wir sicher sein, dass alle notwendigen Abhängigkeiten erfüllt sind und der Apache-Server sowohl installiert als auch anschließend verwendet werden kann.

`emerge` listet an dieser Stelle ausschließlich die Pakete auf, die *noch nicht* installiert sind und die es entsprechend noch installieren müsste, damit der Apache funktioniert.

Einen besseren Überblick über die Hierarchie der Abhängigkeiten erhalten wir mit der Option `--tree` bzw. `-t`, die das obige Listing entsprechend den Abhängigkeiten als Baum darstellt:

```
gentoo ~ # emerge -pvt net-www/apache

These are the packages that would be merged, in reverse order:

Calculating dependencies... done!
[ebuild  N    ] net-www/apache-2.0.58-r2  USE="apache2 ldap ssl -debug -
doc -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mpm-
worker (-selinux) -static-modules -threads" 4,652 kB
[ebuild  N    ]  dev-libs/apr-util-0.9.12  USE="berkdb gdbm ldap" 0 kB
[ebuild  N    ]  app-misc/mime-types-5  0 kB
[ebuild  N    ]  dev-libs/apr-0.9.12  USE="ipv6 -urandom" 0 kB
[ebuild  N    ]  net-nds/openldap-2.3.30-r2  USE="berkdb crypt gdbm ipv6
 perl readline ssl tcpd -debug -kerberos -minimal -odbc -overlays -samba
 -sasl (-selinux) -slp -smbkrb5passwd" 0 kB

Total: 5 packages (5 new), Size of downloads: 4,652 kB
```

Die Pakete sind entsprechend den Abhängigkeiten eingerückt. So benötigt, wie oben bereits gesehen, `net-www/apache` in direkter Abhängigkeit die Bibliothek `apr-util`, also findet sich `dev-libs/apr-util` einfach eingerückt unterhalb von `net-www/apache`. Würde `dev-libs/apr-util` selbst wiederum Pakete benötigen, die `emerge` noch installieren müsste, befänden sich die entsprechenden Einträge zweifach eingerückt unterhalb von `dev-libs/apr-util`.

Bisweilen wird diese Logik durch besondere Typen der Abhängigkeit unterbrochen. Das gilt z. B. für `app-admin/perl-cleaner`, das als Abhängigkeit von `dev-lang/perl` installiert wird, seinerseits jedoch ebenfalls von Perl abhängt. Eine solche zirkuläre Abhängigkeit lässt sich nicht darstellen, und in diesen Fällen bildet `emerge` den Baum stellenweise doppelt ab. Die entstehende Hierarchie ist also mehr eine grobe Orientierung als ein exaktes Abbild der komplexen Abhängigkeiten.

Will man die gesamten Abhängigkeiten des Apache-Servers anzeigen, kann man den Befehl `emerge -pv net-www/apache` um `--emptytree` (oder `-e`) erweitern. `emerge` nimmt dann an, das wir noch gar keine Pakete installiert haben, und zeigt den gesamten Baum notwendiger Pakete.

```
gentoo ~ # emerge -epv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] sys-devel/gnuconfig-20060702  0 kB
[ebuild   R   ] dev-libs/expat-1.95.8  USE="-test" 0 kB
[ebuild   R   ] virtual/libintl-0  0 kB
[ebuild  N    ] dev-libs/apr-0.9.12  USE="ipv6 -urandom" 0 kB
[ebuild   R   ] sys-libs/zlib-1.2.3-r1  USE="-build" 0 kB
[ebuild   R   ] virtual/libiconv-0  0 kB
[ebuild  N    ] app-misc/mime-types-5  0 kB
[ebuild   R   ] sys-devel/autoconf-wrapper-4-r3  0 kB
[ebuild   R   ] sys-devel/automake-wrapper-3-r1  0 kB
[ebuild   R   ] sys-apps/tcp-wrappers-7.6-r8  USE="ipv6" 0 kB
[ebuild   R   ] sys-devel/gettext-0.16.1  USE="nls -doc -emacs -nocxx" 0
 kB
[ebuild   R   ] sys-apps/diffutils-2.8.7-r1  USE="nls -static" 0 kB
[ebuild   R   ] sys-apps/findutils-4.3.2-r1  USE="nls (-selinux) -static
" 0 kB
[ebuild   R   ] sys-devel/m4-1.4.7  USE="nls" 0 kB
[ebuild   R   ] sys-devel/binutils-config-1.9-r3  0 kB
[ebuild   R   ] sys-devel/binutils-2.16.1-r3  USE="nls -multislot -multi
target -test -vanilla" 0 kB
[ebuild   R   ] sys-libs/db-4.3.29-r2  USE="-bootstrap -doc -java -nocxx
 -tcl -test" 0 kB
[ebuild   R   ] sys-libs/gdbm-1.8.3-r3  USE="berkdb" 0 kB
[ebuild   R   ] sys-devel/libperl-5.8.8-r1  USE="berkdb gdbm -debug -ith
reads" 0 kB
[ebuild   R   ] dev-lang/perl-5.8.8-r2  USE="berkdb gdbm -build -debug -
doc -ithreads -perlsuid" 0 kB
[ebuild   R   ] dev-libs/openssl-0.9.8d  USE="zlib -bindist -emacs -sse2
 -test" 0 kB
[ebuild   R   ] dev-perl/Locale-gettext-1.05  0 kB
[ebuild   R   ] perl-core/Test-Harness-2.64  0 kB
[ebuild   R   ] perl-core/PodParser-1.35  0 kB
[ebuild   R   ] sys-apps/help2man-1.36.4  USE="nls" 0 kB
[ebuild   R   ] app-misc/ca-certificates-20061027.2  0 kB
```

```
[ebuild   R   ] sys-libs/ncurses-5.5-r3  USE="gpm unicode -bootstrap -bu
ild -debug -doc -minimal -nocxx -trace" 0 kB
[ebuild   R   ] app-shells/bash-3.1_p17  USE="nls -afs -bashlogger -vani
lla" 0 kB
[ebuild   R   ] sys-apps/texinfo-4.8-r5  USE="nls -build -static" 0 kB
[ebuild   R   ] sys-libs/gpm-1.20.1-r5  USE="(-selinux)" 0 kB
[ebuild   R   ] sys-devel/autoconf-2.61  USE="-emacs" 0 kB
[ebuild   R   ] sys-libs/readline-5.1_p4  0 kB
[ebuild   R   ] app-admin/perl-cleaner-1.04.3  0 kB
[ebuild   R   ] sys-devel/automake-1.10  0 kB
[ebuild   R   ] sys-devel/libtool-1.5.22  0 kB
[ebuild  N    ] net-nds/openldap-2.3.30-r2  USE="berkdb crypt gdbm ipv6
perl readline ssl tcpd -debug -kerberos -minimal -odbc -overlays -samba
-sasl (-selinux) -slp -smbkrb5passwd" 0 kB
[ebuild  N    ] dev-libs/apr-util-0.9.12  USE="berkdb gdbm ldap" 0 kB
[ebuild  N    ] net-www/apache-2.0.58-r2  USE="apache2 ldap ssl -debug -
doc -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mpm-
worker (-selinux) -static-modules -threads" 4,652 kB

Total: 38 packages (5 new, 33 reinstalls), Size of downloads: 4,652 kB
```

Die meisten der hier angegebenen Pakete sind grundlegende Elemente eines Linux-Systems und darum schon in der anfänglich installierten Stage enthalten.

Es ist selten sinnvoll, die Option `--emptytree` zu verwenden, ohne gleichzeitig `--pretend` und `--verbose` zu spezifizieren, da man die bereits installierten Pakete nicht nochmals zu kompilieren braucht. Damit dient das Flag `-e` also eher Informationszwecken. Im Normalfall würden wir hier nur die fünf notwendigen Pakete für den Apache-Server installieren und nicht nochmals alle schon installierten Pakete.

4.2.3 Die Installation durchführen

Würden wir also zum ursprünglichen Befehl `emerge -pv net-www/apache` zurück gehen und das Flag `--pretend` entfernen, würde `emerge` mit der Installation der Pakete beginnen. Da diese Kombination aus einem hypothetischen Lauf mit der Option `-p` und dem tatsächlichen Installationsvorgang ohne diese Option häufig vorkommt, lassen sich beide Vorgänge mit dem Flag `--ask` (bzw. `-a`) kombinieren. Die Option `-a` ersetzt das Flag `-p` folgendermaßen:

```
gentoo ~ # emerge -av net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-nds/openldap-2.3.30-r2  USE="berkdb crypt gdbm ipv6
```

```
perl readline ssl tcpd -debug -kerberos -minimal -odbc -overlays -samba
-sasl (-selinux) -slp -smbkrb5passwd" 0 kB
[ebuild  N    ] dev-libs/apr-0.9.12  USE="ipv6 -urandom" 0 kB
[ebuild  N    ] app-misc/mime-types-5  0 kB
[ebuild  N    ] dev-libs/apr-util-0.9.12  USE="berkdb gdbm ldap" 0 kB
[ebuild  N    ] net-www/apache-2.0.58-r2  USE="apache2 ldap ssl -debug -
doc -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mpm-
worker (-selinux) -static-modules -threads" 4,652 kB

Total: 5 packages (5 new), Size of downloads: 4,652 kB

Would you like to merge these packages? [Yes/No]
```

Die Ausgabe ist dieselbe wie oben, aber am Ende bricht `emerge` nicht ab, sondern fragt, ob wir die Installation jetzt genau so durchführen möchten.

Wer mag, führt die Installation hier mit `Yes` durch und springt dann zu Kapitel 4.4 ab Seite 108. Wir wollen uns aber an dieser Stelle noch ein paar ausgefallenere `emerge`-Optionen ansehen, auf die man aber auch gut zu einem späteren Zeitpunkt zurück kommen kann.

Hinweis für Systeme mit Netzwerkanbindung

Der Quellcode für den Apache-Server ist auf der LiveDVD nicht vorhanden. Sie brauchen also eine funktionierende Netzwerkverbindung, um das Paket hier zu installieren.

Sollte es, aus welchem Grund auch immer, zu einem Abbruch der Installation kommen, bevor `emerge net-www/apache` erfolgreich war, kann man den abgebrochenen Vorgang übrigens wieder mit `--resume` aufnehmen:

```
gentoo ~ # emerge --resume
Calculating dependencies... done!
*** Resuming merge...
...
```

4.3 Fortgeschrittene Installationsmöglichkeiten

`emerge` bietet einige Funktionen für „besondere Situationen“. Angenommen, wir hätten uns an dieser Stelle dafür entschieden, Apache zu installieren, aber es fehlte die Zeit, die Pakete wirklich zu kompilieren, dann könnten wir zumindest schon einmal den Quellcode herunterladen und die eigentliche Installation auf später verschieben. Dieser Fall mag beispielsweise bei einem Laptop eintreffen, mit dem man zu einem späteren Zeitpunkt

keine schnelle Netzanbindung, dafür aber Zeit für den eigentlichen Installationsprozess hat.

4.3.1 Quellcode herunterladen

Um die Aktion von `emerge` auf das Herunterladen der Quellen zu beschränken, fügen wir dem Aufruf die Option `--fetchonly` (bzw. `-f`) hinzu. Der folgende Aufruf würde also nur das Quellarchiv herunterladen und im Verzeichnis `/usr/portage/distfiles` platzieren:

```
gentoo ~ # emerge -fav net-www/apache

These are the packages that would be fetched, in order:

Calculating dependencies... done!
[ebuild  N    ] net-nds/openldap-2.3.30-r2  USE="berkdb crypt gdbm ipv6
perl readline ssl tcpd -debug -kerberos -minimal -odbc -overlays -samba
-sasl (-selinux) -slp -smbkrb5passwd" 0 kB
[ebuild  N    ] dev-libs/apr-0.9.12  USE="ipv6 -urandom" 0 kB
[ebuild  N    ] app-misc/mime-types-5  0 kB
[ebuild  N    ] dev-libs/apr-util-0.9.12  USE="berkdb gdbm ldap" 0 kB
[ebuild  N    ] net-www/apache-2.0.58-r2  USE="apache2 ldap ssl -debug -
doc -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mpm-
worker (-selinux) -static-modules -threads" 4,652 kB

Total: 5 packages (5 new), Size of downloads: 4,652 kB

Would you like to fetch the source files for these packages? [Yes/No] No
```

Die Option `--fetchonly` lässt sich also auch problemlos mit dem `--ask`-Flag verbinden.

4.3.2 Sonderbehandlung der Abhängigkeiten

Manchmal ist es sinnvoll, schon einmal alle Abhängigkeiten aufzulösen, ohne jedoch das eigentliche Paket zu installieren. Vielleicht möchte man sich erst einmal die Dokumentation der Software genauer ansehen, bevor man sie in den aktiven Betrieb nimmt, und alle notwendigen Vorbereitungen treffen. Das ist über die Option `--onlydeps` oder `-o` möglich. Im Apache-Beispiel würden also alle angegebenen Pakete bis auf den eigentlichen Apache-Server installiert.

```
gentoo ~ # emerge -pvo net-www/apache

These are the packages that would be merged, in order:
```

```
Calculating dependencies... done!
[ebuild  N    ] net-nds/openldap-2.3.30-r2  USE="berkdb crypt gdbm ipv6
perl readline ssl tcpd -debug -kerberos -minimal -odbc -overlays -samba
-sasl (-selinux) -slp -smbkrb5passwd" 0 kB
[ebuild  N    ] dev-libs/apr-0.9.12  USE="ipv6 -urandom" 0 kB
[ebuild  N    ] app-misc/mime-types-5  0 kB
[ebuild  N    ] dev-libs/apr-util-0.9.12  USE="berkdb gdbm ldap" 0 kB

Total: 4 packages (4 new), Size of downloads: 0 kB
```

Genau der umgekehrte Fall, also die Installation einer Software, ohne sich um deren Abhängigkeiten zu kümmern, lässt sich mit dem Flag `--nodeps` (bzw. `-O`) bewerkstelligen:

```
gentoo ~ # emerge -pvO net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-www/apache-2.0.58-r2  USE="apache2 ldap ssl -debug -
doc -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mpm-
worker (-selinux) -static-modules -threads" 4,652 kB

Total: 1 package (1 new), Size of downloads: 4,652 kB
```

Diese Vorgehensweise ist allerdings grob fahrlässig und sollte nie notwendig sein, denn sind nicht alle Abhängigkeiten erfüllt, besteht schon bei der Installation ein hohes Risiko, dass der Vorgang erfolglos abbricht. In seltenen Fällen kann diese Option bei einer fehlerhaften Paketdefinition helfen, das Paket trotzdem zu installieren. Allerdings umgeht man so den eigentlichen Zweck des Paketmanagementsystems. Die korrekte Lösung wäre immer, die Paketdefinition so zu korrigieren, das wir nicht zu solch drastischen Maßnahmen greifen müssen.

4.3.3 Paketkonfiguration

An dieser Stelle verlassen wir das Apache-Paket für eine Weile. Ist es noch nicht installiert, sollte man dies nun mit `emerge net-www/apache` nachholen, da wir im Folgenden von dessen Verfügbarkeit ausgehen.

Es bleibt eine weitere zentrale Eigenschaft von `emerge` zu klären, die jedoch nur bei wenigen Paketen zum Tragen kommt: die abschließende Konfiguration der Pakete. Für die meisten Pakete erfolgt die Konfiguration rein manuell, wobei `emerge` die entsprechenden Instruktionen am Ende der Installation ausgibt (siehe auch Kapitel 6.3.4). In einigen Fällen wird allerdings eine automatisierte Alternative angeboten, beispielsweise beim MySQL-Server, den wir darum jetzt installieren werden. Wir fügen diesmal die Option

`--quiet` oder auch `-q` hinzu, um die von Portage ausgegebenen Informationen zu reduzieren:

```
gentoo ~ # emerge -aq =dev-db/mysql-5.0.26-r2
[ebuild  N    ] perl-core/Sys-Syslog-0.18
[ebuild  N    ] sys-apps/ed-0.2-r6
[ebuild  N    ] dev-db/mysql-init-scripts-1.2
[ebuild  N    ] virtual/perl-Storable-2.15
[ebuild  N    ] dev-perl/Net-Daemon-0.39
[ebuild  N    ] dev-db/mysql-5.0.26-r2
[ebuild  N    ] dev-perl/PlRPC-0.2018
[ebuild  N    ] virtual/perl-Sys-Syslog-0.18
[ebuild  N    ] virtual/mysql-5.0
[ebuild  N    ] dev-perl/DBI-1.53
[ebuild  N    ] dev-perl/DBD-mysql-3.0008

Would you like to merge these packages? [Yes/No] Yes
```

Da `emerge` bei der Installation allerdings die Ausgaben des Kompilierens nicht unterdrückt, ist der Effekt von `--quiet` nur begrenzt effektiv.

Wir haben hier auch eine spezifische Version ausgewählt (5.0.26-r2). Würden das nicht tun, treffen wir auf einen Fehler bei der neueren Version[1] und das wollen wir Ihnen hier ersparen.

Hinweis für Systeme mit Netzwerkanbindung

Wenn sie ihr System bereits aktualisiert haben, trifft das nicht mehr zu. Das Problem existierte, als die LiveDVD erstellt wurde und ist mittlerweile behoben.

Wir bestätigen diesmal mit `Yes` und installieren damit MySQL. Die Installation terminiert mit einem Hinweis auf den Befehl zur ersten Konfiguration der Datenbank:

```
 *
 * You might want to run:
 * "emerge --config =dev-db/mysql-5.0.26-r2"
 * if this is a new install.
 *
```

Da wir MySQL wirklich zum ersten Mal installieren, folgen wir den Instruktionen und nutzen die Konfigurationsoption `--config`, die uns `emerge` für diesen Fall anbietet:

[1] `http://bugs.gentoo.org/show_bug.cgi?id=178460`

```
gentoo ~ # emerge --config =dev-db/mysql-5.0.26-r2

Configuring pkg...

 * MySQL DATADIR is /var/lib/mysql
 * Previous datadir found, it's YOUR job to change
 * ownership and take care of it
 * Creating the mysql database and setting proper
 * permissions on it ...
 * Insert a password for the mysql 'root' user
 * Avoid ["'_%] characters in the password
    >meingeheimespasswort
 * Retype the password
    >meingeheimespasswort
. * Loading "zoneinfo", this step may require a few seconds ...
 * Stopping the server ...
 * Done
```

In dem folgenden Dialog werden wir nur dazu aufgefordert, das Root-Passwort für die Datenbank festzulegen. Die Konfigurationsroutine übernimmt es dann, die initialen Datenbankstrukturen so vorzubereiten, dass die Datenbank vollständig einsetzbar ist.

4.3.4 Virtuelle Pakete

Wenn Sie sich die oben abgebildete, anfängliche Ausgabe bei der MySQL-Installation genauer angesehen haben, dann ist Ihnen vielleicht aufgefallen, dass wir gleich zwei Pakete mit dem Namen `mysql` installiert haben. Eines in der Kategorie `dev-db`, welches wir auch offensichtlich installieren wollten.

Aber `emerge` hat noch ein zweites `mysql`-Paket, nämlich aus der Kategorie `virtual`, installiert. Pakete dieser Kategorie sind etwas Besonderes. Wir wollen nicht allzu sehr auf den eigentlichen Mechanismus dieser Pakete eingehen, sondern nur ihren Zweck erläutern.

Es gibt Situationen, in denen verschiedene Pakete exakt die gleiche Funktion ausüben können. Im Fall von MySQL ist das etwas schwieriger zu erklären, deshalb ziehen wir einmal die Sprache Java als Beispiel heran. Wir können alle Java-Programme über die Software von Sun (`dev-java/sun-jdk`) laufen lassen. Es gibt aber genauso die Möglichkeit, eine freie Variante (z. B. `dev-java/blackdown-jdk`) zu verwenden. Das ist für sich genommen erst einmal kein Problem. Wenn wir die freie Variante bevorzugen, dann installieren wir diese eben.

Wie aber können wir nun für ein Java-Programm festlegen, dass es zwingend eine installierte Java-Umgebung braucht? Wir könnten sagen, dass es nur von `dev-java/sun-jdk` abhängt. Das ist aber nicht ganz wahr, schließ-

lich würde das Programm auch mit `dev-java/blackdown-jdk` zusammenarbeiten. Zwingend zu verlangen, dass `dev-java/sun-jdk` installiert ist, ist keine Option, da wir den Benutzer bevormunden würden und Gentoo genau das – wenn irgend möglich – zu vermeiden sucht.

Und genau hier kommen die *virtuellen* Pakete ins Spiel. Sie sind sozusagen eine Art Weiche für verschiedene Pakete, die aber die gleiche Funktionalität bieten. So gibt es also z. B. `virtual/jdk`, das verschiedene Alternativen für die Java-Entwicklungsumgebung abstrahiert.

Ein Java-Programm deklariert also nur, dass es von `virtual/jdk` abhängt. So ist es dem Benutzer überlassen, welches Paket er letztlich wählt. Hat er `dev-java/blackdown-jdk` bereits vorher installiert, betrachtet Portage die `virtual/jdk`-Abhängigkeit als erfüllt und wird keine weitere Java-Entwicklungsumgebung installieren.

Haben wir zum Zeitpunkt der Installation eines virtuellen Paketes noch keine solche Wahl getroffen, entscheidet sich Portage automatisch für ein Standardpaket. Im Fall von Java wird dies `dev-java/sun-jdk` sein.

4.3.5 Pakete aus dem System entfernen

Zuletzt bleibt noch eine weitere Option von `emerge` zu nennen, die wir an dieser Stelle aber nicht wirklich nutzen: `--unmerge`. Diese Aktion dient dem Deinstallieren von Paketen und wird ebenfalls häufig mit der Option `--ask` verknüpft, damit es eine Warnstufe gibt, bevor Portage sich tatsächlich daran macht, die Software zu entfernen.

Um es auszuprobieren, können wir versuchen, Apache wieder zu deinstallieren, beantworten die abschließende Frage dann allerdings mit `No`:

```
gentoo ~ # emerge --unmerge --ask net-www/apache

>>> These are the packages that would be unmerged:

 net-www/apache
    selected: 2.0.58-r2
   protected: none
     omitted: none

>>> 'Selected' packages are slated for removal.
>>> 'Protected' and 'omitted' packages will not be removed.

Would you like to unmerge these packages? [Yes/No] No

Quitting.
```

Bei der Verwendung von `--unmerge` sollte man natürlich Vorsicht walten lassen und wirklich nur Programme entfernen, die man mit Sicherheit nicht

mehr braucht. Deinstallieren wir zentrale Werkzeuge wie `gcc` oder `glibc` mit Absicht oder aus Versehen, legen wir damit das eigene System erst einmal vollständig lahm. Auch in solchen Extremfällen lässt sich Gentoo bzw. ein Linux-System im Allgemeinen reparieren. Aber es ist eine zeitraubende Operation, die man sich ersparen sollte.

Wie weiter oben in Kapitel 4.2.2 gezeigt, installiert `emerge` bei der Apache-Installation einige zusätzliche Pakete aufgrund von Abhängigkeiten im System. Entfernen wir Apache hier mit `--unmerge`, entfernt Portage wirklich nur `net-www/apache`. Wollen wir hingegen alle nicht mehr benötigten Pakete entfernen, die aufgrund von Abhängigkeiten zusätzlich installiert wurden, hätten wir nach dem Entfernen von `net-www/apache` die Möglichkeit, die Option `--depclean` anzuwenden, auch wenn wir von ihrem Einsatz dringend abraten:

```
gentoo ~ # emerge --depclean --ask

** WARNING ** Depclean may break link level dependencies.  Thus, it is
** WARNING ** recommended to use a tool such as ‘revdep-rebuild‘ (from
** WARNING ** app-portage/gentoolkit) in order to detect such breakage.
** WARNING **
** WARNING ** Also study the list of packages to be cleaned for any obvi
ous
** WARNING ** mistakes. Packages that are part of the world set will alw
ays
** WARNING ** be kept.  They can be manually added to this set with
** WARNING ** ‘emerge --noreplace <atom>‘.  Packages that are listed in
** WARNING ** package.provided (see portage(5)) will be removed by
** WARNING ** depclean, even if they are part of the world set.
** WARNING **
** WARNING ** As a safety measure, depclean will not remove any packages
** WARNING ** unless *all* required dependencies have been resolved.  As
 a
** WARNING ** consequence, it is often necessary to run
** WARNING ** ‘emerge --update --newuse --deep world‘ prior to depclean.

Calculating dependencies... done!


>>> These are the packages that would be unmerged:

 app-misc/mime-types
    selected: 5
   protected: none
     omitted: none

 dev-libs/apr
    selected: 0.9.12
   protected: none
     omitted: none
```

```
 dev-libs/apr-util
    selected: 0.9.12
   protected: none
     omitted: none

 net-nds/openldap
    selected: 2.3.30-r2
   protected: none
     omitted: none

>>> 'Selected' packages are slated for removal.
>>> 'Protected' and 'omitted' packages will not be removed.

Would you like to unmerge these packages? [Yes/No] No
```

Portage durchläuft mit dieser Option alle installierten Pakete und testet, welche nur als abhängige Pakete installiert wurden und von keinem anderen installierten Paket mehr benötigt werden. Diese hängt `emerge` der Liste der zu löschenden Elemente an.

Allerdings ist hier Vorsicht angesagt: Portage kann nicht garantieren, dass bei dieser Suche wirklich alle Abhängigkeiten korrekt identifiziert werden, vor allem dann, wenn sich noch andere Systemparameter verändert haben. Letztlich kann es passieren, dass man sich mit dieser Option das System zerstört und es, wie bereits erwähnt, umständlicher Rettungsaktionen bedarf. Die Option `--ask` ist hier also Pflicht, und man sollte sich die zu deinstallierenden Pakete genau anschauen, bevor man mit `Yes` antwortet – was wir hier ohnehin nicht tun, da wir den Apache-Server noch benötigen.

4.4 Paketpräfix

Kommen wir zurück von den ausgefalleneren `emerge`-Optionen zu den grundlegenden Konzepten hinter Portage und damit auch `emerge`. Wir bleiben allerdings vorerst noch auf der Kommandozeile und schauen uns den Paketnamen genauer an, bevor wir dann auf die Konfigurationseinstellungen im Hintergrund eingehen.

4.4.1 Paketversionen

Wir haben es uns bisher einfach gemacht und nur den Paketnamen für die Installation verwendet. Während der Installation haben wir unter 1.7.1 aber auch gesehen, dass es noch etwas komplizierter geht und wir die Versionsnummern mit in die Bezeichnung aufnehmen können.

Schauen wir uns dazu noch einmal den Inhalt des Apache-Paketverzeichnisses an und überprüfen, welche Version derzeit installiert ist:

```
gentoo ~ # ls -la /usr/portage/net-www/apache/
insgesamt 188
drwxr-xr-x  3 root root  4096 31. Jan 2007  .
drwxr-xr-x 50 root root  4096 14. Mär 19:21 ..
-rw-r--r--  1 root root  8891 11. Jan 2007  apache-1.3.34-r14.ebuild
-rw-r--r--  1 root root  9066 11. Jan 2007  apache-1.3.37.ebuild
-rw-r--r--  1 root root 13828 28. Jan 2007  apache-2.0.58-r2.ebuild
-rw-r--r--  1 root root 14075 28. Jan 2007  apache-2.0.59-r2.ebuild
-rw-r--r--  1 root root 14160 28. Jan 2007  apache-2.2.4.ebuild
-rw-r--r--  1 root root 88130 31. Jan 2007  ChangeLog
drwxr-xr-x  2 root root  4096  8. Mär 20:58 files
-rw-r--r--  1 root root  6983 31. Jan 2007  Manifest
-rw-r--r--  1 root root   551 16. Jan 2007  metadata.xml
gentoo ~ # emerge -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] net-www/apache-2.0.58-r2  USE="apache2 ldap ssl -debug -
doc -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mpm-
worker (-selinux) -static-modules -threads" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB
```

Offensichtlich ist derzeit Version `2.0.58-r2` installiert; nicht die neueste Version (`2.2.4`), aber auch nicht die älteste (`1.3.34-r14`).

4.4.2 Bestimmte Paketversionen installieren

Nehmen wir an, wir möchten eine andere Apache-Version installieren, weil z. B. die neueste Version, obwohl als „stabil“ markiert (siehe dazu auch Abschnitt 5.3.1), noch einen Fehler hat, der uns betrifft und uns dazu zwingt, doch lieber eine ältere Version zu verwenden. Oder wir müssen mit einem Upgrade auf neuere Versionen auch umfangreiche Konfigurationen oder Datenbestände anpassen, so dass wir aus diesem Grund vielleicht zunächst noch der älteren Variante den Vorzug geben, um die Aktualisierung erst zu einem späteren Zeitpunkt durchzuführen.

Beim Apache-Webserver ist das auch der Grund, warum noch die alten `apache-1`-Versionen verfügbar sind. Es gibt noch zahlreiche Webserver, bei denen der Aufwand der Migration in keinem Verhältnis zum Nutzen steht; so bleibt man lieber bei der alten Version und aktualisiert in kleinen Schritten.

Nehmen wir hier also einmal an, wir möchten über unseren Webserver einige ältere Web-Applikation anbieten, die nur mit den Apache-1-Versionen kompatibel sind. Mit dem Gleichheitszeichen (`=`) als Paket-Präfix können wir die gewünschte Version exakt vorgeben:

```
gentoo ~ # emerge -pv =net-www/apache-1.3.34-r14

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-www/gentoo-webroot-default-0.2  USE="-no-htdocs" 65
kB
[ebuild  N    ] dev-libs/mm-1.3.0  221 kB
[ebuild  NS   ] net-www/apache-1.3.34-r14  USE="pam ssl -doc -lingerd (-
selinux) -static-modules" 3,239 kB
[ebuild  N    ] net-www/mod_ssl-2.8.25-r10  0 kB

Total: 4 packages (3 new, 1 in new slot), Size of downloads: 3,524 kB
```

Das Ergebnis mag ein wenig überraschen: Zum einen fordert `emerge` plötzlich weitere Pakete an. Das lässt sich aber damit erklären, dass unterschiedliche Versionen einer Software in ihren Eigenschaften und damit auch den Abhängigkeiten variieren. `net-www/apache-1.3.34-r14` hat beispielsweise auch ganz andere USE-Flags (siehe auch das nächste Kapitel 5.1).

Zum anderen zeigt aber das `N` in `[ebuild NS ]` an, dass wir den älteren Apache-Server *neu* installieren werden. Der Apache-Server liegt hier in einer speziellen Kategorie von Paketen, die mehrfach auf einem System in unterschiedlichen *Slots* installiert sein können. Genauer erklären wir das beim Aktualisieren von Paketen in Kapitel 10.4.1 ab Seite 228. Hier sei nur gesagt, dass es so möglich wird, gleichzeitig sowohl `apache-1` als auch `apache-2` auf einem System zu betreiben. Auch die Migration eines alten Webservers wird dadurch erleichtert. Bei den meisten Paketen würde die vorangestellte Information jedoch `[ebuild D ]` enthalten. Das `D` zeigt dann ein *Downgrade*, also den Wechsel zu einer niedrigeren Version an.

Übrigens wird die Paketbezeichnung, die man `emerge` mit auf den Weg gibt, von Portage *Atom* genannt. Sollte man gelegentlich auf die Meldung `xyz is not a valid atom` von `emerge` stoßen, soll einem das sagen, dass in der Paketbezeichnung ein Fehler steckt, dass wir z. B. – was sehr häufig passiert – das voranzustellende Gleichheitszeichen vor einem Paketnamen vergessen haben:

```
gentoo ~ # emerge -pv net-www/apache-1.3.34-r14

These are the packages that would be merged, in order:

Calculating dependencies /

!!! 'net-www/apache-1.3.34-r14' is not a valid package atom.
!!! Please check ebuild(5) for full details.
!!! (Did you specify a version but forget to prefix with '='?)
```

Hier gibt Portage allerdings auch einen deutlichen Hinweis.

4.4.3 Ausgefallenere Versionsauswahl

Wenn es ein Präfix = gibt, liegt es nahe, dass auch < und > sowie <= und >= zulässige Angaben sind. In der im letzten Abschnitt beschriebenen Situation können wir z. B. auch angeben, dass wir „irgendeine“ Version kleiner als `apache-2` installieren wollen:

```
gentoo ~ # emerge -pv "<net-www/apache-2"

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-www/gentoo-webroot-default-0.2  USE="-no-htdocs" 65
kB
[ebuild  N    ] dev-libs/mm-1.3.0  221 kB
[ebuild  NS   ] net-www/apache-1.3.34-r14  USE="pam ssl -doc -lingerd (-
selinux) -static-modules" 3,239 kB
[ebuild  N    ] net-www/mod_ssl-2.8.25-r10  0 kB

Total: 4 packages (3 new, 1 in new slot), Size of downloads: 3,524 kB
```

Hier sind die Anführungszeichen um den Paketnamen Pflicht, denn andernfalls interpretiert die Bash-Kommandozeile das <-Zeichen als Character mit besonderer Funktion. Wir erzielen also das gleiche Ergebnis wie zuvor, müssen aber zuvor nicht umständlich die exakte Version ermitteln.

Statt des < hätten wir auch ein Suffix verwenden können – das von der Kommandozeile bekannte Sternchen (`*`) und würden damit irgendein Paket, dessen Version mit 1 beginnt, installieren. Im Ergebnis also das gleiche wie `<net-www/apache-2`:

```
gentoo ~ # emerge -pv "=net-www/apache-1*"
```

Es ist eher selten, dass wir wirklich auf einer älteren Version verharren möchten. Die umgekehrte Situation, dass z. B. ein Fehler die aktuell stabile wie auch ältere Versionen betrifft, so dass wir auf die noch instabile, aber in dieser Hinsicht korrigierte Version setzen, ist wahrscheinlicher. Oder die jüngere Variante bietet neue Eigenschaften, die wir gerne nutzen würden. Auch in diesem Fall würden wir die neuere Version bevorzugen.

Wollen wir eine höhere Version des Apache-Servers installieren, bedienen wir uns des > oder des >=, also:

```
gentoo ~ # emerge -pv ">=net-www/apache-2.2.0"

These are the packages that would be merged, in order:

Calculating dependencies |
!!! All ebuilds that could satisfy ">=net-www/apache-2.2.0" have been ma
```

```
sked.
!!! One of the following masked packages is required to complete your re
quest:
- net-www/apache-2.2.4 (masked by: package.mask, ~x86 keyword)
# Michael Stewart <vericgar@gentoo.org> (03 Feb 2006)
# Mask for testing of new Apache 2.2 version


For more information, see MASKED PACKAGES section in the emerge man page
or refer to the Gentoo Handbook.
```

Die Aktion schlägt in diesem Falle fehl, da Portage uns davor bewahren möchte, eine potentiell instabile Version zu installieren. Wie wir solche Restriktionen umgehen, sehen wir im nächsten Kapitel in Abschnitt 5.5.

Kapitel 5

Hinter den Kulissen von emerge

Bislang ging es um einen Überblick darüber, wie man `emerge` über die Kommandozeile steuert und Pakete installiert bzw. wieder entfernt. Wir können das Paketmanagementsystem darüber hinaus aber auch unabhängig davon grundlegend konfigurieren, wobei die Datei `/etc/make.conf` eine zentrale Rolle spielt. Weitere Konfigurationsmöglichkeiten finden sich im Verzeichnis `/etc/portage`.

Hier soll es zunächst um die Variablen `USE` und `ACCEPT_KEYWORDS` aus der Datei `/etc/make.conf` bzw. aus den Dateien ähnlicher Funktion in `/etc/portage` (`/etc/portage/package.*`) gehen. Beide Variablen berühren fundamentale Konzepte von Portage, weshalb wir hier die Hintergründe genauer ausführen und uns dann im nächsten Kapitel mit den übrigen Konfigurationsmöglichkeiten in `/etc/make.conf` beschäftigen.

Obwohl dieses Kapitel eher theoretisch ist, empfiehlt es sich nicht, es zu überspringen. Wer noch keine Erfahrung mit Gentoo hat, wird hier die grundlegenden Unterschiede zu anderen Distributionen und die komfortable Arbeit mit der Paketverwaltung Portage kennen lernen.

5.1 USE-Flags

Portage installiert Software auf Basis des Quellcodes. Verglichen mit vorkompilierten Distributionen sind die Einflussmöglichkeiten damit zahlreicher und flexibler. Beim Bauen etwa von RPM- oder Debian-Paketen verfolgt man üblicherweise die Strategie, alle möglichen Features der entsprechenden Software zu aktivieren, so dass der User bei Bedarf darauf zurückgreifen kann.

In den meisten Fällen benötigt aber nicht jeder Nutzer jede dieser oft umfassenden Funktionalitäten einer Software oder Bibliothek. Steht beispielsweise fest, dass wir LDAP nicht einsetzen wollen, dann ist es wenig sinnvoll, dies zu aktivieren und die entstehenden Binaries dadurch zu vergrößern. In anderen Fällen reduziert der Verzicht auf bestimmte Funktionalität z. B. die Zahl der Konfigurationsdateien, so dass der Nutzer leichteren Zugang zur Software findet.

Wie sich bestimmte Eigenschaften an- oder abschalten lassen, unterscheidet sich von Software zu Software. In vielen Fällen gibt es einen passenden Switch für den `configure`-Aufruf, der dem Kompilieren der Software vorausgeht, in anderen Fällen muss der Quellcode entsprechend gepatcht werden oder man steht vor der Herausforderung, Dateien hinzuzufügen bzw. zu löschen. Gentoo kapselt diese Vorgänge so, dass der Nutzer sie unabhängig von der Software einheitlich handhaben kann.

Entsprechend kann jedes Gentoo-Paket so genannte *USE-Flags* definieren und damit signalisieren, dass es bestimmte optionale Eigenschaften bietet, die der Nutzer zur Compile-Zeit aktivieren kann. Wie `emerge` diese Flags in Paket-Installationsanweisungen umsetzt, spielt aus User-Sicht keine Rolle – um die nötigen Schritte beim Kompilieren oder Installieren kümmern sich die Gentoo-Paketbauer, im Distributionsslang (wie bei Debian) „Entwickler“ genannt.

Möchte man ein neues Paket installieren, stellt sich also meist die Frage, welche besonderen Features die Software unterstützt und welche von diesen man aktivieren möchte. Diese Auskunft gibt `emerge`:[1]:

```
gentoo ~ # USE="hardened" emerge -pv sys-libs/glibc

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] sys-libs/glibc-2.5  USE="hardened* nls nptl nptlonly -bu
ild -glibc-compat20 -glibc-omitfp (-multilib) -profile (-selinux)" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB
```

[1] Den Zusatz `USE="hardened"` erklären wir etwas später, in Kapitel 5.1.3 ab Seite 124

Mit einer Reihe Sonderzeichen und einem Farbcode liefert `emerge` Informationen zu den (de)aktivierbaren Eigenschaften des Pakets; denen, die aufgrund der aktuellen Systemkonfiguration bei der Installation des Pakets nicht aktiviert sind, ist ein Minuszeichen vorangestellt und `emerge` stellt sie in blauer Schrift dar (im obigen Beispiel `-build`, `-glibc-omitfp`, `-hardened`, `-multilib`, `-profile`, `-selinux`). Aktive USE-Flags sind rot eingefärbt (`nls`, `nptl`, `nptlonly`) und tragen keine weitere Markierung.

Bei Paketen, die wir schon einmal installiert haben und die wir nun erneut einspielen, oder bei Software, die wir auf die neueste Version bringen, stehen grüne USE-Flags für solche, die sich im Vergleich zur letzten Installation geändert haben, also von uns entweder aktiviert oder deaktiviert wurden (`hardened*`). Sie sind am Ende mit einem Sternchen markiert. Bei installierten Paketen, für die es ein Update gibt, markiert `emerge` USE-Flags mit einem Prozentzeichen, sofern sie in der neuen Paketversion neu hinzugekommen sind. Außerdem sind sie gelb ausgezeichnet. Im obigen Beispiel handelt sich nicht um ein Update, und entsprechend finden sich hier auch keine neuen Eigenschaften. Über den Farbcode sieht der User sofort, welche neuen USE-Flags für ihn möglicherweise interessant sind.

`emerge` sortiert die USE-Flags in der Anzeige entsprechend ihrem Status (aktiviert/deaktiviert). Die aktivierten Flags stehen vorn. Wer die Flags alphabetisch sortiert haben möchte, verwendet die Option `--alphabetical`:

```
gentoo ~ # USE="hardened" emerge -pv sys-libs/glibc --alphabetical

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] sys-libs/glibc-2.5  USE="-build -glibc-compat20 -glibc-o
mitfp hardened* (-multilib) nls nptl nptlonly -profile (-selinux)" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB
```

Generell unterscheidet man zwischen globalen und lokalen USE-Flags. Die globalen schalten Features an oder aus, die sich auf mehrere Pakete identisch auswirken, während lokale USE-Flags sich nur auf einzelne Pakete beziehen. Eine Beschreibung der globalen USE-Flags findet sich in der Datei `/usr/portage/profiles/use.desc`, während Beschreibungen der lokalen Varianten in `/usr/portage/profiles/use.local.desc` stehen. Die Beschreibung des `ldap`-USE-Flags liefert also z. B. der Befehl:

```
gentoo ~ # grep "^ldap" /usr/portage/profiles/use.*
/usr/portage/profiles/use.desc:ldap - Adds LDAP support (Lightweight Dir
ectory Access Protocol)
```

Ähnliche Auskunft erteilt das Tool `euses`. Um nicht auf `grep` zurückgreifen zu müssen, installieren wir es hier mit

```
gentoo ~ # emerge -av app-portage/euses

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-portage/euses-2.5.4  16 kB

Total: 1 package (1 new), Size of downloads: 16 kB

Would you like to merge these packages? [Yes/No] Yes
```

Hinweis für Systeme mit Netzwerkanbindung

Wir benötigen wieder eine funktionierende Netzwerkverbindung, um das Werkzeug herunterzuladen.

So verkürzt sich der obige Aufruf auf `euses ldap`, wobei das Programm nun allerdings alle USE-Flags anzeigt, in denen die Zeichenkette `ldap` enthalten ist:

```
gentoo ~ # euses ldap
ldap - Adds LDAP support (Lightweight Directory Access Protocol)
dev-lang/php:ldap-sasl - Add SASL support for the PHP LDAP extension
net-fs/samba:ldapsam - Enables samba 2.2 ldap support (default passwd ba
ckend: ldapsam_compat)
```

`euses` zeigt hier das globale USE-Flag `ldap` und seine Beschreibung, listet aber darüber hinaus auch zwei lokale USE-Flags, die die Zeichenkette `ldap` enthalten. Dass die beiden letzten lokale USE-Flags sind, lässt sich daran erkennen, dass `euses` sie inklusive des Pakets nennt, für das sie existieren. Das USE-Flag `ldap` ist als globales USE-Flag eben nicht auf einzelne Pakete beschränkt, darum fehlt hier eine entsprechende Angabe.

Noch einfacher lässt sich die Erklärung der USE-Flags eigentlich durch das `equery`-Skript erhalten. Allerdings schauen wir uns dieses Werkzeug erst in Kapitel 11.1.1 an. Wie sich `equery` zur Anzeige der USE-Flags nutzen lässt, findet sich dann auf Seite 249.

Die für die Installation zu benutzenden USE-Flags kann man durch die Variable `USE` festlegen, die sich auf vier verschiedene Arten modifizieren lässt. Auf der untersten Ebene definiert das anfänglich gewählte Profil (siehe Kapitel 1.8 ab Seite 44) ein Basis-Set an Eigenschaften, das global für alle Pakete gilt. Wir wollen hier nicht detailliert auf die Profile eingehen; einen Teil der Grundkonfiguration für die `USE`-Variable findet man z. B. in der Datei `/usr/portage/profiles/default-linux/x86/2007.0/make.defaults`:

```
gentoo ~ # cat /usr/portage/profiles/default-linux/x86/2007.0/make.defa\
> ults
# Copyright 1999-2007 Gentoo Foundation
# Distributed under the terms of the GNU General Public License v2
# $Header: /var/cvsroot/gentoo-x86/profiles/default-linux/x86/dev/2007.0
/make.defaults,v 1.4 2007/02/12 16:18:07 wolf31o2 Exp $

# We build stage1 against this
STAGE1_USE="nptl nptlonly unicode"

# These USE flags are what is common between the various sub-profiles. S
tages 2
# and 3 are built against these, so be careful what you add.
USE="acl cups gdbm gpm libg++ nptl nptlonly unicode"
```

Wir sehen hier nur einen Teil der durch das Profil aktivierten USE-Flags, da Profile hierarchisch organisiert sind und so auch die darüber liegende Ebene `/usr/portage/profiles/default-linux/x86/make.defaults` ins Profil einbezogen wird. Genauer beschreiben wir Profile weiter im Kapitel 5.2 ab Seite 127.

5.1.1 euse

Die im Profil vordefinierten Flags lassen sich über das Tool `euse` (nicht zu verwechseln mit `euses`!) auslesen. Das Programm gehört zu einer wichtigen Sammlung spezieller Gentoo-Tools, die auf keinem Gentoo-System fehlen sollte. Sie ist im Paket `app-portage/gentoolkit` enthalten, das wir an dieser Stelle folgendermaßen installieren:

```
gentoo ~ # emerge -av app-portage/gentoolkit

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-portage/gentoolkit-0.2.3  0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
```

Unter Angabe der Optionen `--info` (bzw. `-i`; Informationen zu allen USE-Flags ausgeben), `--global` (bzw. `-g`; Informationen auf globale USE-Flags beschränken) und `--active` (bzw. `-a`; nur aktivierte Flags zeigen) zeigt `euse` an, welche USE-Flags derzeit aktiv sind:

```
gentoo ~ # euse -i -g -a
acl                [+  D ]
```

```
apache2             [+ C  ]
berkdb              [+  D ]
cracklib            [+  D ]
crypt               [+  D ]
cups                [+  D ]
dri                 [+  D ]
fortran             [+  D ]
gdbm                [+  D ]
gpm                 [+  D ]
iconv               [+  D ]
ipv6                [+  D ]
ladspa              [+    ]
ldap                [+ C  ]
libg++              [+  D ]
mysql               [+ C  ]
ncurses             [+  D ]
nls                 [+  D ]
nptl                [+  D ]
pam                 [+  D ]
pcre                [+  D ]
perl                [+  D ]
python              [+  D ]
readline            [+  D ]
session             [+  D ]
spl                 [+  D ]
ssl                 [+  D ]
tcpd                [+  D ]
unicode             [+  D ]
v4l                 [+    ]
xml                 [+ C  ]
zlib                [+  D ]
```

Das Pluszeichen verrät, dass das entsprechende Flag aktiv ist, das große `D` besagt, dass diese Einstellung aus dem gewählten Gentoo-Profil stammt. Darauf aufbauend gibt es drei weitere Möglichkeiten, USE-Flags für das gesamte System oder auch spezifisch für einzelne Pakete zu (de)aktivieren.

Systemweite USE-Flags legt man in der Datei `/etc/make.conf` in der Variablen `USE` fest. Wenn man einen Desktop-Rechner installiert, wird man hier z. B. das Flag `X` hinzufügen und damit für viele Pakete die Unterstützung der grafischen Oberfläche aktivieren. Auf Server-Systemen hingegen empfiehlt es sich meistens, dem gleichen Flag ein Minus voranzustellen, um es global zu deaktivieren. Sinnvolle globale USE-Flags für ein LAMP-Server-System haben wir schon während der Installation auf Seite 45 gesetzt, indem wir die folgende Zeile in `/etc/make.conf` eingetragen haben:

```
USE="-X apache2 ldap mysql xml"
```

Das ist auch der Grund dafür, dass wir im Listing weiter oben einige USE-Flags mit der Markierung C sehen. euse zeigt an, dass wir das entsprechende USE-Flag in der Datei /etc/make.conf aktiviert haben.

In dieser Zeile haben wir das USE-Flag X explizit deaktiviert. Diese explizite Angabe ist notwendig, da X im übergeordneten Profil als aktiv gesetzt ist. In den meisten Fällen ist diese Art, das Grundprofil zu modifizieren, die sinnvollste. Sollte man jedoch eine Maschine installieren wollen, die bei den USE-Flags sehr stark von den Standardeinstellungen abweicht, können sich einige explizit deaktivierte USE-Flags ansammeln.

In diesen Fällen kann es einfacher sein, der Variable USE das spezielle USE-Flag -* voranzustellen (USE="-* ..."). Nun ignoriert emerge alle USE-Flag-Einstellungen aus dem Profil. Von dieser Option sollte man jedoch nur Gebrauch machen, wenn man genau weiß, was man tut. Die Einstellungen im Profil wurden schließlich nicht grundlos erstellt. Im schlimmsten Fall kann -* dazu führen, dass System-Paketen wichtige Eigenschaften für den Betrieb fehlen und man seine Maschine somit lahm legt.

Anstatt den Wert der oben angegebenen USE-Variable manuell in die Datei /etc/make.conf zu schreiben, lässt sich diese Aufgabe auch dem vorher installierten euse-Tool übertragen. Aktivieren wir probehalber einmal das USE-Flag X mit der Option --enable (bzw. -E) für ein Desktop-System und betrachten das Ergebnis in /etc/make.conf:

```
gentoo ~ # euse -E X
/etc/make.conf was modified, a backup copy has been placed at /etc/make.
conf.euse_backup
gentoo ~ # grep USE /etc/make.conf
USE="apache2 ldap mysql xml X"
```

Die neu aktivierten USE-Flags markiert euse jetzt mit dem Buchstaben C und informiert darüber, dass wir sie in der Datei /etc/make.conf gesetzt haben:

```
gentoo ~ # euse -i -g -a
...
unicode             [+  D ]
v4l                 [+    ]
X                   [+ C  ]
xml                 [+ C  ]
zlib                [+  D ]
```

Um die Einstellung wieder zurück zu nehmen, benutzen wir euse mit der -Option -D (bzw. --disable):

```
gentoo ~ # euse -D X
/etc/make.conf was modified, a backup copy has been placed at /etc/make.
conf.euse_backup
```

```
gentoo ~ # grep USE /etc/make.conf
USE="apache2 ldap mysql xml -X"
```

Wie wir sehen, verbleibt das USE-Flag mit vorangestelltem Minuszeichen in der Datei `/etc/make.conf`. Damit deaktivieren wir das USE-Flag explizit für alle Pakete. Wenn wir keine Aussage zu dem USE-Flag treffen möchten, es also in der Variable `USE` nicht auftauchen soll, können wir `euse` mit der Option `-P` (bzw. `--prune`) verwenden:

```
gentoo ~ # euse -P X
/etc/make.conf was modified, a backup copy has been placed at /etc/make.
conf.euse_backup
gentoo ~ # grep USE /etc/make.conf
USE="apache2 ldap mysql xml"
```

Man sollte nur die allgemein gehaltenen Eigenschaften auf diese Weise global in `/etc/make.conf` konfigurieren.

Die USE-Flags legen grundsätzlich *nicht* fest, dass ein entsprechend benanntes Paket im System installiert wird. `apache2` in `/etc/make.conf` festzulegen führt also nicht automatisch dazu, dass der Apache-Server installiert wird. Lediglich bei der Installation von Paketen, die irgendeine Unterstützung für den Apache-Server, für MySQL oder PHP mitbringen, sorgen die so gesetzten USE-Flags dafür, dass `emerge` diese Unterstützung im Paket aktiviert.

Vielfach muss das entsprechende Paket (also z. B. `net-nds/openldap` für das USE-Flag `ldap`) installiert sein, damit diese Art Interaktion zwischen den Paketen auch tatsächlich möglich ist. So können die USE-Flags also auch die Abhängigkeiten einzelner Pakete beeinflussen. Das haben wir im vorigen Kapitel auf Seite 98 schon kurz angerissen, als die Installation des Apache-Servers auch die Installation von OpenLDAP verlangte.

USE-Flags wirken sich auf die Installation der Pakete aus, d. h. dass Änderungen an USE-Flags erst dann wirksam werden, wenn wir ein Paket mit veränderten USE-Flags neu installieren. Für diesen Fall bietet `emerge` die Option `--newuse` (bzw. `-N`), über die sich Pakete ansprechen lassen, deren USE-Flags sich im Vergleich zur letzten Installation geändert haben:

```
gentoo ~ # euse -D ldap
/etc/make.conf was modified, a backup copy has been placed at /etc/make.
conf.euse_backup
gentoo ~ # emerge --newuse -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] net-www/apache-2.0.58-r2  USE="apache2 ssl -debug -doc -
ldap* -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mp
```

```
m-worker (-selinux) -static-modules -threads" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB
gentoo ~ # euse -E ldap
/etc/make.conf was modified, a backup copy has been placed at /etc/make.
conf.euse_backup
gentoo ~ # emerge --newuse -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!

Total: 0 packages, Size of downloads: 0 kB
```

Im ersten Fall, wenn wir das USE-Flag `ldap` entfernt haben, merkt Portage, dass sich die Bedingungen für die Apache-Installation geändert haben, und bietet die erneute Installation von Apache mit veränderten USE-Flags an. Sobald wir das USE-Flag wieder aktiviert haben, ist alles beim Alten und `emerge` zeigt an, dass es nichts zu tun gibt.

5.1.2 Paketspezifische USE-Flags

Schauen wir uns noch einmal die paketspezifischen USE-Flags des Apache-Servers an:

```
gentoo ~ # emerge -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] net-www/apache-2.0.58-r2  USE="apache2 ldap ssl -debug -
doc -mpm-itk -mpm-leader -mpm-peruser -mpm-prefork -mpm-threadpool -mpm-
worker (-selinux) -static-modules -threads" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB
```

Wir sehen hier einige USE-Flags mit der Bezeichnung `mpm-*`. Um herauszufinden, wofür diese zuständig sind, verwenden wir wieder `euses`:

```
gentoo ~ # euses mpm-
net-www/apache:mpm-event - (experimental) Event MPM - a varient of the w
orker MPM that tries to solve the keep alive problem - requires epoll su
pport and kernel 2.6
net-www/apache:mpm-itk - (experimental) Itk MPM - child processes have s
eperate user/group ids
net-www/apache:mpm-leader - (experimental) Leader MPM - leaders/follower
s varient of worker MPM
net-www/apache:mpm-peruser - (experimental) Peruser MPM - child processe
```

```
s have seperate user/group ids
net-www/apache:mpm-prefork - Prefork MPM - non-threaded, forking MPM - s
imiliar manner to Apache 1.3
net-www/apache:mpm-threadpool - (experimental) Threadpool MPM - keeps po
ol of idle threads to handle requests
net-www/apache:mpm-worker - Worker MPM - hybrid multi-process multi-thre
ad MPM
```

Wie am Paketnamen ersichtlich, haben wir es hier mit paketspezifischen bzw. *lokalen* USE-Flags zu tun, die nur für das Paket `net-www/apache` gültig sind. Die `mpm-*`-Flags dienen beim Apache-Server dazu, die verschiedenen Multi-Processing-Module (MPM) zu (de)aktivieren. Jedes der Module bestimmt das Verhalten des Webservers bei den zahlreichen gleichzeitigen Anfragen.

Wir haben zwar an dieser Stelle gar kein MPM-Modul ausgewählt, aber das Apache-Paket ist so definiert, dass es eigenständig `mpm-prefork` selektiert, wenn der Benutzer keine Wahl trifft. Nehmen wir an, wir wollten den Apache über den Prefork-Modus hinaus mit dem MPM-Worker-Modul betreiben und dabei auch Threads (ein Verfahren, ein Programm in verschiedene Aktivitätsabläufe zu splitten) verwenden. Dann wollen wir speziell für das Apache-Paket die USE-Flags `mpm-prefork` (Modul MPM-Prefork kompilieren), `mpm-worker` (Modul MPM-Worker kompilieren) und `threads` („threaded"-Modus unterstützen) aktivieren.

Paketspezifische Flags (de)aktiviert man in `/etc/portage/package.use`, indem man pro Zeile ein Paket und die zugehörigen USE-Flags angibt. Als Trennzeichen zwischen den einzelnen Flags dient je ein Leerzeichen:

```
net-www/apache mpm-prefork mpm-worker threads
```

Die angegebenen USE-Flags lassen sich auch auf bestimmte Versionen beschränken, indem man dem Paketnamen ein Präfix (siehe Kapitel 4.4) voranstellt und die gewünschte Version anhängt (etwa `<net-www/apache-2*`). Dies sollte jedoch nur selten notwendig sein.

Bei einem Eintrag ist dieses Verfahren noch recht übersichtlich. Wer aber nach längerem Betrieb des Systems zweihundert Einträge gesammelt hat wünscht sich möglicherweise etwas mehr Übersicht. Statt eine einzelne Datei `/etc/portage/package.use` zu verwenden, lässt sich dieser Pfad auch als Verzeichnis anlegen.

Alle darin vorhandenen Dateien fügt Portage intern automatisch zusammen. So lassen sich die Einstellungen thematisch besser gruppieren und verwalten, indem man z. B. `/etc/portage/package.use/web-server` dafür nutzt, USE-Flags spezifisch für den Webserver festzulegen, während man in `/etc/portage/package.use/desktop` die USE-Flags für Desktop-relevante Pakete sammelt.

5.1.3 flagedit

Auch die Datei `/etc/portage/package.use` lässt sich statt mit dem Texteditor mit einem spezialisierten Tool bearbeiten. Dafür installieren wir das Paket `app-portage/flagedit`:

```
gentoo ~ # emerge -av app-portage/flagedit

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] dev-perl/Array-RefElem-1.00  2 kB
[ebuild  N    ] virtual/perl-MIME-Base64-3.07  0 kB
[ebuild  N    ] dev-perl/XML-Parser-2.34  0 kB
[ebuild  N    ] dev-perl/DelimMatch-1.06  9 kB
[ebuild  N    ] dev-perl/Data-DumpXML-1.06  8 kB
[ebuild  N    ] dev-util/libconf-0.40.00  USE="xml" 314 kB
[ebuild  N    ] app-portage/flagedit-0.0.5  5 kB

Total: 7 packages (7 new), Size of downloads: 337 kB

Would you like to merge these packages? [Yes/No] Yes
```

Hinweis für Systeme mit Netzwerkanbindung

Auch hier sind wieder nicht alle Quellpakete auf der DVD verfügbar und wir benötigen die Verbindung ins Netz, um das Paket installieren zu können.

Ähnlich wie `euse` erlaubt es uns, systemweite USE-Flags zu aktivieren (mit vorangestelltem Pluszeichen), zu deaktivieren (mit vorangestelltem Minuszeichen) oder sie vollständig zu entfernen (mit vorangestelltem Prozentzeichen):

```
gentoo ~ # flagedit +X
gentoo ~ # grep USE /etc/make.conf
USE="apache2 ldap mysql xml X"
gentoo ~ # flagedit %X
gentoo ~ # grep USE /etc/make.conf
USE="apache2 ldap mysql xml"
gentoo ~ # flagedit -X
gentoo ~ # grep USE /etc/make.conf
USE="apache2 ldap mysql xml -X"
```

Als Pfad nimmt `flagedit` per Default `/etc/make.conf` an, aber diese Standardeinstellung lässt sich mit der Option `--make-conf-file` beeinflussen.

Um paketspezifische statt systemweite USE-Flags über `flagedit` zu definieren, stellt man den Paketnamen vor die zu (de)aktivierenden USE-Flags (Mit dem Switch `--show` listet `flagedit` die aktuell aktiven USE-Flags für das angegebene Paket.):

```
gentoo ~ # flagedit net-www/apache +mpm-prefork +mpm-worker +threads
gentoo ~ # flagedit net-www/apache --show
mpm-prefork mpm-worker threads
```

`flagedit` modifiziert standardmäßig `/etc/portage/package.use`, was sich aber über die Option `--package-file` ändern lässt. Das ist besonders dann notwendig, wenn man `/etc/portage/package.use` als Verzeichnis verwendet.

Wenn wir die paketspezifischen Einstellungen verwenden, sollte sich das auch auf unsere Installation auswirken:

```
gentoo ~ # emerge -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] net-www/apache-2.0.58-r2  USE="apache2 ldap mpm-prefork*
 mpm-worker* ssl threads* -debug -doc -mpm-itk -mpm-leader -mpm-peruser
-mpm-threadpool (-selinux) -static-modules" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB
```

Bei diesem Aufruf sind die drei neu aktivierten USE-Flags (`mpm-worker`, `mpm-prefork` und `threads`) grün markiert und mit einem Sternchen versehen: Sie sind also in unserer Installation nicht aktiviert und würden sich erst bei einer erneuten Installation auswirken.

Drei Varianten, über die sich USE-Flags setzen lassen (Profil, `make.conf`, `package.use`) haben wir jetzt kennen gelernt. Zuletzt lassen sich USE-Flags aber auch über die Umgebungsvariable `USE` setzen:

```
gentoo ~ # USE="-mpm-prefork mpm-peruser" emerge -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] net-www/apache-2.0.58-r2  USE="apache2 ldap mpm-peruser*
 mpm-worker* ssl threads* -debug -doc -mpm-itk -mpm-leader -mpm-prefork
-mpm-threadpool (-selinux) -static-modules" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB
```

Diese Methode hat einen gravierenden Nachteil: Installiert man das Paket in dieser Form, indem man das der Variablendefinition nachgestellte `emerge`-Kommando ohne `-pv` aufruft, so gelten die angegebenen USE-Flags nur für diese Installation. Beim nächsten Update vergisst man meist, die Variable mit anzugeben und spielt somit eine neue Version auf, der die benötigte Funktionalität fehlt. Für Testzwecke eignet sich die Spezifikation von USE-Flags per Umgebung gut, bei positivem Ergebnis überträgt man sie besser in die Datei `/etc/portage/package.use`.

5.1.4 Spezielle USE-Flags

Es gibt einige Situationen, in denen das USE-Flag-Konzept etwas einengend wirkt. Schauen wir uns zur Demonstration einmal an, was `emerge` liefert, wenn wir den Firefox-Browser installieren möchten (wir verwenden hier die `--nodeps`-Option (bzw. `-O`; siehe Seite 103), da wir nicht gleich ein ganzes Desktop-System installieren wollen):

```
gentoo ~ # emerge -pvO www-client/mozilla-firefox

These are the packages that would be merged, in order:

[ebuild  N    ] www-client/mozilla-firefox-2.0.0.3  USE="ipv6 -bindist -
debug -filepicker -gnome -java -mozdevelop -moznopango -restrict-javascr
ipt -xforms -xinerama -xprint" LINGUAS="-af -ar -be -bg -ca -cs -da -de
-el -en_GB -es -es_AR -es_ES -eu -fi -fr -fy -fy_NL -ga -ga_IE -gu -gu_I
N -he -hu -it -ja -ka -ko -ku -lt -mk -mn -nb -nb_NO -nl -nn -nn_NO -pa
-pa_IN -pl -pt -pt_BR -pt_PT -ru -sk -sl -sv -sv_SE -tr -zh -zh_CN -zh_T
W" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB
```

`emerge` zeigt hier plötzlich nicht nur die Variable `USE` an, sondern gleich dahinter, in ähnlichem Format, die Variable `LINGUAS`. Die darin gelisteten Einträge lassen schon erahnen, worum es sich hier handelt: Die Unterstützung verschiedener Sprachen durch dieses Pakets. Gerade populäre Software liegt oftmals in verschiedenen Sprachen vor. Nun ist es im deutschsprachigen Raum aber selten sinnvoll, die Unterstützung zum Beispiel für das Japanische mit zu installieren.

Es geht also um eine bei der Installation relevante Entscheidung und entsprechend sollte es USE-Flags geben, die dem Benutzer die Auswahl erlauben, welche Sprachen er installiert haben möchte. Es wäre möglich gewesen, die verschiedenen Sprachen durch globale USE-Flags zu repräsentieren, aber die Übersichtlichkeit bei den bestehenden globalen USE-Flags hätte stark gelitten.

Das gleiche Problem besteht auch bei Paketen, die eine Vielzahl unterschiedlicher Hardware unterstützen und die entsprechenden Treiber in-

stallieren. Warum sollte man den Grafikkarten-Treiber für Radeon-Karten installieren, wenn man eine Nvidia-Karte verwendet? Die Vielzahl verschiedener Grafikkarten-Treiber des X-Servers gaben den eigentlichen Ausschlag für das Konzept der *erweiterten USE-Flags*.

Anfänglich waren nur `LINGUAS` (für Sprachen), `VIDEO_DEVICES` (für Grafikkarten) und `INPUT_DEVICES` (für Eingabegeräte) definiert. Mittlerweile hat das Konzept aber stärkere Verbreitung gefunden; in `/usr/portage/local/gentoo/gentoo-portage/profiles/base/make.defaults` stehen unter `USE_EXPAND` alle derzeit gültigen erweiterten USE-Flags.

Die erweiterten USE-Flags funktionieren eigentlich genau wie `USE` selbst auch. Wir können sie in `/etc/make.conf` festlegen oder auch auf der Kommandozeile angeben. Die Einträge in `/etc/make.conf` können beispielsweise folgendermaßen aussehen:

```
LINGUAS="de en"
INPUT_DEVICES="keyboard mouse"
VIDEO_CARDS="nvidia"
```

Unter `LINGUAS` sollte man im deutschen Sprachraum wenigstens `de` und `en` angeben. Als grundlegende `INPUT_DEVICES` bieten die meisten Systeme eine Tastatur (`keyboard`) und eine Maus (`mouse`). Bleibt noch die Grafikkarte unter `VIDEO_DEVICES` aufzuführen. Die meisten werden sich hier zwischen `nvidia` und `radeon` entscheiden müssen.

Für eine Server-Maschine, wie wir sie hier installieren, spielen diese Einstellungen jedoch eine geringere Rolle, da bisher vor allem die Desktop-orientierten Pakete (und primär der X-Server `x11-base/xorg-server`) diese Mechanismen unterstützen.

Abschließend seien noch zwei besondere USE-Flags erwähnt, die man als Benutzer jedoch nicht setzen kann bzw. darf. Zum einen das USE-Flag `test`, zum anderen die *architekturspezifischen USE-Flag*.

Bringt ein Paket automatisierte Testroutinen mit, dann lassen sich diese während der Installation durchlaufen, indem man der Variablen `FEATURES` (mehr dazu im nächsten Kapitel ab Seite 150) in `/etc/make.conf` die Option `test` hinzufügt. In diesem Fall aktiviert Portage intern automatisch das USE-Flag `test`. Diesen automatischen Ablauf dürfen wir nicht stören, indem wir es in `/etc/make.conf` eintragen.

Gleiches gilt für die *architekturspezifischen USE-Flags*. Portage aktiviert automatisch ein USE-Flag entsprechend dem gewählten Profil (z. B. `x86` für das x86-Profil). Auch diese USE-Flags dürfen Benutzer nicht zur `USE`-Variablen hinzufügen.

5.2 Profile

Wir haben uns schon während der Installation in Kapitel 1.8 kurz mit den Grundlagen der *Profile* in `/usr/portage/profiles` beschäftigt. Während es bei der Installation erst einmal nur um die verschiedenen Verzeichnisse und die dadurch repräsentierten Profile ging, haben wir uns in den vorangegangenen Abschnitten gelegentlich auf den Inhalt dieser Verzeichnisse (z. B. `make.defaults` oder die Datei `use.desc`) bezogen.

Wir wollen an dieser Stelle einen kurzen Überblick über die wichtigsten Funktionen von `/usr/portage/profiles` geben. Direkt im Verzeichnis `/usr/portage/profiles` liegen einige Dateien, die globale Informationen für Portage liefern.

`use.desc`, `use.local.desc`
: Diese Dateien liefern eine kurze Beschreibung der USE-Flags. Beide haben wir schon auf Seite 115 kennen gelernt.

`arch.list`
: Eine Liste der von Gentoo unterstützten Architekturen.

`categories`
: Die Liste der gültigen Kategorien des Portage-Baumes.

`ChangeLog`
: Die von den Entwicklern in diesem Verzeichnis vorgenommenen Änderungen.

`info_*`
: Definiert, welche Informationen `emerge --info` ausgibt. Siehe auch Seite 371.

`package.mask`
: Die Liste maskierter Pakete. Siehe weiter unten Kapitel 5.5.

`profiles.desc`
: Eine Übersicht über die verfügbaren Profile und deren Stabilitätsgrad.

`repo_name`
: Der Name des Repositories, das unter `/usr/portage` liegt. Das ist eigentlich immer der Portage-Baum und der trägt den Namen `gentoo`. Diese Datei im Verzeichnis `profiles` spielt erst eine Rolle, wenn wir uns im Kapitel 14 mit Overlays beschäftigen.

`thirdpartymirrors`
: Enthält eine Liste von häufig verwendeten Servern für das Herunterladen von Quellpaketen.

Das Verzeichnis `desc` enthält die Beschreibung der erweiterten USE-Flags, die wir weiter oben auf Seite 126 angesprochen haben. Das Verzeichnis `updates` definiert spezielle Operationen, die Portage durchführen muss, wenn Pakete z. B. die Kategorie wechseln. Die darin enthaltenen Dateien sind nach Quartalen sortiert.

Die übrigen Verzeichnisse definieren – abgesehen von `obsolete`, das aber tatsächlich obsolet ist – Gentoo-Profile. Ein Profil-Verzeichnis enthält im Normalfall eine Sammlung von Sub-Profilen und Dateien, die Grundeigenschaften des Profils definieren.

Wir wollen hier nur auf die wichtigsten Dateien eingehen:

`parent`
: Ist dieser Eintrag vorhanden, handelt es sich um ein Sub-Profil und `parent` verweist dann auf das Eltern-Profil. Dieser Mechanismus erlaubt hierarchische Profile und macht die Definition derselben effizient. Ein Sub-Profil erbt alle Einstellungen des Eltern-Profils und kann diese abwandeln.

`make.defaults`
: Definiert die Standardeinstellungen für Variablen, die der Benutzer mit Hilfe von `/etc/make.conf` verändern kann.

`packages`
: Die Pakete, die für dieses Profil zwingend installiert sein müssen.

`package.use`
: USE-Flags, die in diesem Profil für die entsprechenden Pakete dringend empfohlen werden oder schlicht notwendig sind.

`package.use.mask`
: USE-Flags, die in diesem Profil für die entsprechenden Pakete nicht verwendet werden können oder möglicherweise zu Fehlfunktionen führen.

`use.force`
: USE-Flags, die in diesem Profil zwingend notwendig sind.

`use.mask`
: USE-Flags, die in diesem Profil nicht verwendet werden können oder möglicherweise zu Fehlfunktionen führen.

`virtuals`
: Definiert für virtuelle Pakete (siehe Kapitel 4.3.4) die Standardauswahl für dieses Profil.

5.2.1 Das Profil mit eselect auswählen

Das zu verwendende Profil lässt sich auch mit dem Werkzeug `eselect` auswählen. Genaueres zu dem Programm findet sich in Kapitel 11.4 ab Seite 263. Wir wollen uns hier nur mit dem `profile`-Modul beschäftigen, installieren aber erst einmal `eselect`:

```
gentoo ~ # emerge -av app-admin/eselect

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-admin/eselect-1.0.7  USE="-bash-completion -doc" 0 k
B

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
```

Einen Überblick über die Funktionen des `profile`-Moduls liefert `eselect profile`:

```
gentoo ~ # eselect profile
Usage: eselect profile <action> <options>

Standard actions:
  help                      Display help text
  usage                     Display usage information
  version                   Display version information

Extra actions:
  list                      List available profile symlink targets
  set <target>              Set a new profile symlink target
    target                    Target name or number (from 'list' action)
    --force                   Forcibly set the symlink
  show                      Show the current make.profile symlink
```

`show` zeigt also das derzeit ausgewählte Profil:

```
gentoo ~ # eselect profile show
Current make.profile symlink:
  /usr/portage/profiles/default-linux/x86/2007.0
```

Die für unsere Architektur gültigen Profile liefert der Befehl `list`:

```
gentoo ~ # eselect profile list
Available profile symlink targets:
  [1]   default-linux/x86/2006.1
```

```
[2]   default-linux/x86/no-nptl
[3]   default-linux/x86/no-nptl/2.4
[4]   default-linux/x86/2006.1/desktop
[5]   hardened/x86/2.6
[6]   selinux/x86/2006.1
```

Verwunderlich ist hier, das unser derzeit aktiviertes Profil `2007.0` sich nicht wiederfindet. Denn das `profile`-Module ist in Version 1.0.7 von `eselect` noch fehlerhaft, wir können es hier also nicht wie geplant verwenden. Wenn Sie ihr System später aktualisiert haben, wird die neuere Version korrekt funktionieren.

Sollten wir das Profil wechseln wollen, bedienen wir uns der Option `set`:

```
gentoo ~ # eselect profile set 2
```

`eselect` verändert hier lediglich den `/etc/make.profile`-Symlink. Allerdings wird das Tool zunehmend für verschiedene Gentoo-Konfigurationsaufgaben genutzt und bietet ein einheitliches Interface. Deshalb ist seine Verwendung durchaus zu empfehlen.

5.3 Keywords

Die gleichen Mechanismen, die für die Variable `USE` gelten, kommen auch bei `ACCEPT_KEYWORDS` zum Tragen. Auch hinter den so genannten *Keywords*, steckt ein recht komplexes Konzept, das wir uns nun genauer ansehen wollen.

Keywords liefern die Grundlage, um die Stabilität einzelner Paketversionen zu markieren. Es gibt hier nur zwei Stufen: instabil und stabil. Jede (Prozessor-)Architektur, unter der Gentoo läuft, bietet ein eigenes Keyword, so z. B. `x86` für i*86-basierte Maschinen, `ppc` für die PowerPC-Architektur oder `amd64` für 64-Bit-Maschinen.

5.3.1 Stabil/Instabil

Um anzuzeigen, dass ein Paket für eine entsprechende Architektur als stabil gilt, nehmen die Entwickler das entsprechende Keyword in die Paketdefinition auf. Ist das Paket auf der Plattform installierbar, jedoch noch nicht ausreichend getestet, so fügen sie das entsprechende Keyword mit einer vorangestellten Tilde (also ~x86, ~amd64 etc.) hinzu.

Instabil markierte Pakete sind allerdings nicht nur weniger getestet, sondern Gentoo gibt darüber hinaus für diese Pakete auch keine Sicherheitswarnungen heraus (siehe Seite 253).

Fehlt die Keyword-Angabe in der Paketdefinition, dann hat entweder noch kein Gentoo-Entwickler das Paket auf einer solchen Plattform installiert oder es ist bekannt, dass dieses Paket nicht konfliktfrei auf der entsprechenden Architektur läuft. `emerge` weigert sich in diesen Fällen, das entsprechende Paket zu installieren.

Mit der Variablen `ACCEPT_KEYWORDS` legen wir nun fest, welche Keywords wir für unser System akzeptieren. Wie bei den USE-Flags gibt es vier Ebenen, über die wir `ACCEPT_KEYWORDS` modifizieren können. Die Standardeinstellung findet sich wieder im gewählten Profil. Für die `x86`-Architektur in `/usr/portage/profiles/default-linux/x86/make.defaults`:

```
gentoo ~ # grep KEYWORDS \
> /usr/portage/profiles/default-linux/x86/make.defaults
ACCEPT_KEYWORDS="x86"
```

Für die anderen Architekturen findet sich der Standard-Wert analog im zugehörigen Profil (z. B. `amd64` in `/usr/portage/profiles/default-linux/amd64/make.defaults`).

Auf der zweiten Ebene lässt sich `ACCEPT_KEYWORDS` wieder über die Datei `/etc/make.conf` beeinflussen. Im Normalfall ist kein Wert vorgegeben, dann zählt die Einstellung des Profils.

Um das eigene System nur mit instabilen Paketen zu fahren, ist es möglich, das Keyword mit einer Tilde zu versehen (`ACCEPT_KEYWORDS="~x86"`). Von diesem Vorgehen raten wir jedoch dringend ab. Die instabilen Pakete tragen ihren Namen nicht zu Unrecht und sind vielfach mit Vorsicht zu genießen (siehe auch Seite 253). Ein vollständig mit instabilen Paketen bestücktes System erfreut im Normalfall höchstens die Gentoo-Entwickler, da sie mehr Fehlerberichte erhalten. Für den Nutzer resultieren die möglichen Probleme jedoch sehr leicht in einer hohen Menge Frustration. Auch das erhöhte Sicherheitsrisiko einer solchen Maschine im Netz sollte man bedenken.

Bietet eine neue, noch instabil markierte Version jedoch Eigenschaften, die man dringend benötigt, möchte man diese natürlich auch installieren. Hier wirkt sich die Tatsache, dass Gentoo Pakete auf Basis des Quellcodes installiert, extrem günstig aus: Es ist überhaupt kein Problem, eine im wesentlichen aus stabilen Paketen bestehende Installation mit ein paar instabilen Paketen zu „würzen“. Selbst die zentralen Pakete einer Distribution lassen sich problemlos in der instabilen Variante nutzen, ohne dass man gleich die gesamte Installation in ein instabiles System verwandelt.

Auf einem System, auf dem `ACCEPT_KEYWORDS` in `/etc/make.conf` wie oben geschildert nur stabile Pakete zulässt, verweigert `emerge` die Installation einer derzeit instabil markierten Apache-Version:

```
gentoo ~ # emerge -pv ">=net-www/apache-2.2.0"
```

```
These are the packages that would be merged, in order:

Calculating dependencies /
!!! All ebuilds that could satisfy ‘‘>=net-www/apache-2.2.0’’ have been
masked.
!!! One of the following masked packages is required to complete your
request:
- net-www/apache-2.2.4 (masked by: package.mask, ~x86 keyword)
# Michael Stewart <vericgar@gentoo.org> (03 Feb 2006)
# Mask for testing of new Apache 2.2 version

For more information, see MASKED PACKAGES section in the emerge man page
or refer to the Gentoo Handbook.
```

Um einzelne Pakete in der instabilen Version zu akzeptieren, bietet Portage den gleichen Mechanismus wie für die USE-Flags. Wir können in der Datei `/etc/portage/package.keywords` paketspezifisch Keywords angegeben. Es gilt, was schon für `/etc/portage/package.use` galt: die Datei können wir alternativ auch als Verzeichnis anlegen. In diesem Fall fasst `emerge` alle Dateien innerhalb von `/etc/portage/package.keywords` zusammen.

Das Format für `package.keywords` ist dasselbe wie für `package.use`. Einer Zeile mit der Paketbezeichnung folgen die akzeptierten Keywords für dieses Paket:

```
net-www/apache ~x86
```

Obiges Beispiel würde auf der Maschine die instabile Apache-Version akzeptieren. Bevor man instabile Pakete auf der eigenen Maschine akzeptiert, sollte man sich auch Gedanken um die Möglichen Auswirkungen auf die Sicherheit des eigenen Systems machen. Wir gehen darauf in Kapitel 11.1.2 ab Seite 253 genauer ein.

In vielen Fällen zieht das Akzeptieren eines Paketes in der instabilen Version einen Rattenschwanz instabiler Pakete hinter sich her, da die neuen Paketversionen vielfach auch von Paketen abhängig sind, die noch nicht stabil markiert sind. Die entsprechenden Pakete muss man ebenfalls in die Datei `/etc/portage/package.keywords` aufnehmen, damit `emerge` das eigentliche Paket bereitwillig installiert. Diese Aufgabe kann man sich mit dem Werkzeug `autounmask` erleichtern. Wir beschreiben es weiter unten in Abschnitt 5.6.

Wieder können wir uns des Werkzeugs `flagedit` bedienen, um die paketspezifischen Keywords zu editieren. Ein doppeltes Minus (`--`) teilt dem Tool mit, dass wir Keywords statt USE-Flags editieren möchten:

```
gentoo ~ # flagedit net-www/apache -- +~x86
gentoo ~ # flagedit net-www/apache -- --show
~x86
```

Keywords lassen sich mit dem Prozentzeichen wieder entfernen:

```
gentoo ~ # flagedit net-www/apache -- %
gentoo ~ # flagedit net-www/apache -- --show

gentoo ~ # flagedit net-www/apache -- +~x86
gentoo ~ # flagedit net-www/apache -- --show
~x86
```

Testen wir einmal, wie sich das auf den `emerge`-Aufruf auswirkt:

```
gentoo ~ # emerge -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild     U ] net-www/apache-2.0.59-r2 [2.0.58-r2] USE="apache2 ldap m
pm-prefork* mpm-worker* ssl threads* -debug -doc -mpm-itk -mpm-leader -m
pm-peruser -mpm-threadpool (-selinux) -static-modules" 4,690 kB

Total: 1 package (1 upgrade), Size of downloads: 4,690 kB
```

Es würde hier also zu einem Upgrade kommen, was `emerge` durch das vorne stehende `U` anzeigt. Während `2.0.58-r2` also die derzeit stabile Version darstellt, ist `net-www/apache-2.0.59-r2` noch in der Testphase, und die Entwickler betrachten sie als ungeeignet für Produktivsysteme.

Ähnlich wie `USE` lässt sich auch die Variable `ACCEPT_KEYWORDS` als Umgebungsvariable beim `emerge`-Aufruf verwenden, um instabile Pakete temporär zu akzeptieren:

```
gentoo ~ # ACCEPT_KEYWORDS="~x86" emerge -pv net-www/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild     U ] net-www/apache-2.0.59-r2 [2.0.58-r2] USE="apache2 ldap m
pm-prefork* mpm-worker* ssl threads* -debug -doc -mpm-itk -mpm-leader -m
pm-peruser -mpm-threadpool (-selinux) -static-modules" 4,690 kB

Total: 1 package (1 upgrade), Size of downloads: 4,690 kB
```

Auch hier raten wir allerdings davon ab, diesen Mechanismus zu verwenden. Da die Einstellung nur temporär gesetzt wird, kann man die Tatsache, dass man ein instabiles Paket installiert hat, beim nächsten Update leicht vergessen, und `emerge` wählt dann wieder die stabile Version.

5.4 Ein instabiles Paket auswählen

Wir haben schon davon abgeraten, systemweit instabile Pakete zu installieren. Wann sollte man denn überhaupt ein instabiles Paket wählen?

Es gibt eigentlich nur zwei Gründe, die für ein instabiles Paket sprechen:

- Das instabil markierte Paket hat Eigenschaften, die in der stabilen Version nicht vorhanden sind, die man aber dringend benötigt.
- Das stabil markierte Paket hat einen Fehler, der in der instabilen Version behoben wurde.

In beiden Fällen ist es sinnvoll, sich über die Entscheidung der Gentoo-Entwickler hinwegzusetzen und das instabile Paket zu installieren. Beide oben beschriebenen Situationen setzen voraus, dass man sich schon etwas intensiver mit dem Paket beschäftigt hat. Dann ist es auch deutlich einfacher, im Problemfall eine Lösung zu finden.

Im Idealfall aktiviert man das instabile Keyword auch nur für eine spezifische Version:

```
gentoo ~ # flagedit =net-www/apache-2.0.59-r2 -- +~x86
gentoo ~ # flagedit =net-www/apache-2.0.59-r2 -- --show
~x86
```

Damit ist sichergestellt, dass wir nicht grundsätzlich das instabile Paket installieren, sondern wirklich nur das Paket, an dem wir interessiert sind. Irgendwann markieren die Entwickler diese Paketversion vermutlich als stabil, während eine neue instabile Version zur Verfügung gestellt wird. Haben wir das Keyword versionsspezifisch festgelegt, wechseln wir in diesem Fall nicht automatisch auf das ganz frische, instabile Paket, das uns möglicherweise neue Probleme bringen würde.

Auf keinen Fall sollte man aus dem Wunsch heraus, immer die neueste Version verwenden zu wollen, zu den instabilen Varianten greifen. Das ist zwar eine durchaus menschliche Regung, aber die Vorteile, die einem dann meist ohnehin unbekannt sind, wiegen selten die Nachteile auf. Man muss sich vor Augen führen, dass deutlich mehr Personen die stabilen Pakete verwenden und damit auch testen, als dies bei den instabilen der Fall ist.

Zudem ist man für die Absicherung der instabilen Pakete selbst zuständig. Nur für die stabil markierten Pakete gibt Gentoo Sicherheitswarnungen[2] heraus. Wer also eine instabile Webapplikation installiert, ohne die einschlägigen Sicherheitslisten wie CVE[3] oder auch Secunia[4] regelmäßig auf

[2] http://www.gentoo.org/rdf/en/glsa-index.rdf
[3] http://cve.mitre.org/
[4] http://secunia.com

Sicherheitsprobleme mit der verwendeten Version abzusuchen, der darf sich nicht wundern, wenn Hacker den Server plötzlich zum Versand von Spam-Mails missbrauchen.

Generell lautet also auch hier die Devise, den Spieltrieb zu unterdrücken, wenn man sich nicht die Finger verbrennen will. Solange man sich aber Gedanken um die Auswahl von instabilen Paketen macht, kann man sein Gentoo-System in ausgewählten Bereichen sehr nah an den neuesten Stand der Entwicklung bringen.

Es bleibt vielleicht die Frage, wie Pakete überhaupt die Markierung „stabil“ erhalten und ob man diesen Prozess als Benutzer beeinflussen kann. Grundsätzlich wandert jedes neue Paket und auch jede neue Paketversion schon stabil markierter Pakete erst einmal als instabil markiert in den Portage-Baum. Erst nach einer Testphase, die mindestens 30 Tage betragen muss, können die Entwickler das Paket als stabil markieren. Diese Markierung dürfen allerdings nur so genannte Architektur-Teams vornehmen.

Für jedes Keyword gibt es ein entsprechendes Team an Entwicklern, die auf ihren Systemen ausschließlich stabile Pakete verwenden. Die Team-Mitglieder sind beim *Stabilisieren* eines Paketes (also z. B. dem Verändern des Keywords von `~x86` auf `x86`) verpflichtet, das Paket innerhalb ihrer stabilen Systemumgebung zu testen und zu überprüfen, ob es problemlos funktioniert. So garantiert Gentoo eine hohe Qualität der stabilen Pakete.

Das Stabilisieren der Pakete lässt sich in begrenztem Umfang auch von Gentoo-Benutzern beeinflussen. Man kann eine entsprechende Stabilisierungsanfrage im Gentoo-Bug-Tracker[5] stellen. Bevor man eine solche Anfrage stellt, sollte man allerdings auf drei Dinge achten:

- Das Paket muss sich schon seit mindestens 30 Tagen im instabilen Zustand im Portage-Baum befinden.
- Kein stabiles Paket darf von instabilen Paketen abhängen. D. h. alle Pakete, von denen das zu stabilisierende abhängt, müssen ebenfalls stabil verfügbar sein.
- Es darf im Gentoo-Bug-Tracker keine offenen Bugs in Bezug auf dieses Paket geben.

Missachtet man diese Grundregeln, kann die Antwort der Entwickler etwas harscher ausfallen.

Sobald ein Paket als stabil markiert ist, ist auch das Sicherheitsteam in der Pflicht, eventuelle Sicherheitslücken möglichst schnell zu stopfen und die Nutzer zu informieren.

[5] `http://bugs.gentoo.org`

5.5 Maskierte Pakete

Abgesehen von den vorher besprochenen Keywords gibt es noch einen weiteren Mechanismus, mit dem die Gentoo-Entwickler die Verwendung einzelner Pakete einschränken können. Vor allem in Fällen, in denen Pakete die Stabilität des Gesamtsystems beeinträchtigen könnten oder neuere Versionen sehr starke Veränderungen aufweisen, übernehmen die Entwickler sie häufig in maskierter Form in den Portage-Baum. Hierfür wird das Paket normal verfügbar gemacht, aber in der Datei `/usr/portage/profiles/package.mask` ein Eintrag hinzugefügt, dass die Benutzer die entsprechende Version noch nicht installieren sollen. Dieser Eintrag listet einfach einen entsprechenden Paketnamen:

```
gentoo ~ # grep -B 3 =net-www/apache /usr/portage/profiles/package.mask

# Michael Stewart <vericgar@gentoo.org> (03 Feb 2006)
# Mask for testing of new Apache 2.2 version
>=net-www/apache-2.2.0
```

Mit `-B 3` erhalten wir die drei Kommentarzeilen vor dem eigentlichen Eintrag. Sie liefern eine kurze Begründung, warum das entsprechende Paket maskiert ist.

`emerge` weigert sich aufgrund dieses Eintrag, eine Apache-Version `>=2.2.0` zu installieren. Natürlich lässt sich diese Sperre von Nutzerseite umgehen, andernfalls ergäbe es keinen Sinn, die neue Version überhaupt im Baum verfügbar zu machen. Damit können Nutzer mit dem nötigen Know-how die Software testen und somit beim Beseitigen von Fehlern helfen, bevor die Entwickler die neue Version einer breiteren Masse an Nutzern als instabiles Paket zur Verfügung stellen.

Um die Sperre aufzuheben, kann der Benutzer den gleiche Mechanismus verwenden wie auch schon für die paketspezifischen USE-Flags und Keywords. Innerhalb des Verzeichnisses `/etc/portage` kann man die Datei `package.unmask` angelegen, einzelne Pakete eintragen und somit deren Maskierung aufheben. Auch diese Datei kann man wieder als Verzeichnis anlegen. Jedoch sind nicht viele Pakete maskiert, so dass `package.unmask` nie wirklich viele Einträge enthalten wird und damit selten Gefahr läuft unübersichtlich zu werden.

Da die Entscheidung, ein Paket zu demaskieren, eine Ja/Nein-Entscheidung ist, ist das Format der Datei `package.unmask` nochmal etwas einfacher als das der bisher besprochenen `package.*`-Dateien. Es reicht, den Paketnamen ohne weitere Zusätze anzugeben:

```
gentoo ~ # echo ">=net-www/apache-2.2.0" >> /etc/portage/package.unmask
gentoo ~ # cat /etc/portage/package.unmask
>=net-www/apache-2.2.0
```

Wie zu sehen, können wir auch hier wieder komplexe Paketnamen verwenden und die Version mit angeben. Aufgrund des einfachen Formats gibt es keinen speziellen Editor für die Datei `package.unmask`.

```
gentoo ~ # emerge -pv ">=net-www/apache-2.2.0"

These are the packages that would be merged, in order:

Calculating dependencies /
!!! All ebuilds that could satisfy "=dev-libs/apr-util-1*" have been mas
ked.
!!! One of the following masked packages is required to complete your re
quest:
- dev-libs/apr-util-1.2.8 (masked by: ~x86 keyword)

For more information, see MASKED PACKAGES section in the emerge man page
or refer to the Gentoo Handbook.
(dependency required by "net-www/apache-2.2.4" [ebuild])
```

Offensichtlich können wir hier die neuere Apache-Version trotzdem noch nicht installieren. Es macht sich bemerkbar, was wir auch schon oben auf Seite 132 angesprochen haben: Meist ist ein maskiertes oder instabiles Paket auch abhängig von anderen Paketen, die nur in instabiler Version vorliegen. In diesem Fall zieht die neue Apache-Version glücklicherweise keinen Rattenschwanz an instabilen Paketen nach sich. Es reicht, `=dev-libs/apr-util-1.2.8` in der instabilen Version zu akzeptieren und schon lässt sich `net-www/apache-2.2.4` installieren:

```
gentoo ~ # flagedit =dev-libs/apr-util-1.2.8 -- +~x86
gentoo ~ # emerge -pv ">=net-www/apache-2.2.0"

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  NS   ] dev-libs/apr-1.2.8  USE="ipv6 -debug -urandom" 1,082 kB
[ebuild  N    ] dev-util/pkgconfig-0.20  USE="-hardened" 0 kB
[ebuild  N    ] dev-libs/libpcre-6.6  USE="-doc" 0 kB
[ebuild  NS   ] dev-libs/apr-util-1.2.8  USE="berkdb gdbm ldap -postgres
 -sqlite -sqlite3" 632 kB
[ebuild     U ] net-www/apache-2.2.4 [2.0.58-r2] USE="ldap mpm-prefork*
mpm-worker* ssl threads* -debug -doc -mpm-event% -mpm-peruser -no-suexec
% (-selinux) -static-modules (-apache2%*) (-mpm-itk%) (-mpm-leader%) (-m
pm-threadpool%)" 4,867 kB

Total: 5 packages (1 upgrade, 2 new, 2 in new slots), Size of downloads:
 6,580 kB
```

Wie schon erwähnt, kommen wir gleich noch auf `autounmask` zu sprechen. Es erlaubt, die Kette an benötigten instabilen Paketen automatisiert zu demaskieren.

In manchen Fällen tritt der umgekehrte Fall ein: Man möchte eine neuere Version nicht installieren. Dies kann vor allem bei Produktivsystemen passieren, bei denen ein Software-Update viel Planung voraussetzt oder man größere Datenbestände auf die neue Version migrieren muss. Wie wir auf Seite 110 gesehen haben, kann man `emerge` dazu verwenden, ganz spezifische Paketversionen zu installieren und somit das unerwünschte neue Paket zu umgehen. Aber vielfach aktualisiert man das System global, und in diesen Situationen würde `emerge` vollautomatisch alle neuen Versionen installieren.

Um in solchen Fällen weiterhin die Möglichkeit zu einem komfortablen globalen Update zu haben, lassen sich spezifische Pakete vom Nutzer über die Datei `/etc/portage/package.mask` von der Aktualisierung ausklammern. Damit hat jene Datei also die gleiche Funktion wie `/usr/portage/profiles/package.mask`, nur dass sie im Gegensatz zu dieser direkt vom Nutzer editierbar ist. Das Format stimmt mit dem von `/etc/portage/package.unmask` überein.

5.6 Pakete mit autounmask demaskieren

Nachdem wir weiter oben ausgiebig vor instabilen und maskierten Paketen gewarnt haben, liefern wir gegen Ende des Kapitels noch ein Werkzeug, mit dem sich diese einfacher verwenden lassen: `autounmask`.

Hinweis für Systeme mit Netzwerkanbindung

Das Werkzeug ist noch sehr neu und auf der LiveDVD ist nicht einmal die Paketdefinition vorhanden. Um es zu installieren, müssen Sie also über eine Netzwerkverbindung verfügen *und* ihr System bereits aktualisiert haben.

Da wir im Folgenden von einem aktualisierten System ausgehen, verwenden wir auch nicht mehr `net-www/apache-2.2.0`, sondern `www-servers/apache-2.2.8`, da zum jetzigen Zeitpunkt (Januar 2008) die Version 2.2.8 als instabil maskiert ist.

Wie schon auf Seite 132 erwähnt, hängt ein instabiles bzw. maskiertes Paket häufig von Paketen ab, die ebenfalls nur instabil verfügbar sind. Man kann diese einzeln in `/etc/portage/package.keywords` eintragen oder sie automatisch mit `autounmask` hinzufügen. Für Letzteres müssen wir das Programm aber erst einmal installieren:

```
gentoo ~ # emerge -av app-portage/autounmask

These are the packages that would be merged, in order:
```

```
Calculating dependencies... done!
[ebuild  N    ] perl-core/Scalar-List-Utils-1.19  43 kB
[ebuild  N    ] dev-perl/Term-ANSIColor-1.12  USE="-test" 14 kB
[ebuild  N    ] dev-perl/Compress-Raw-Zlib-2.005  203 kB
[ebuild  N    ] dev-perl/Net-SSLeay-1.30  77 kB
[ebuild  N    ] dev-perl/IO-String-1.08  8 kB
[ebuild  N    ] virtual/perl-Test-Harness-2.64  0 kB
[ebuild  N    ] dev-perl/yaml-0.65  92 kB
[ebuild  N    ] virtual/perl-Scalar-List-Utils-1.19  0 kB
[ebuild  N    ] dev-perl/IO-Socket-SSL-1.12  51 kB
[ebuild  N    ] dev-perl/IO-Compress-Base-2.005  89 kB
[ebuild  N    ] dev-perl/PortageXS-0.02.07  USE="-minimal" 28 kB
[ebuild  N    ] dev-perl/IO-Compress-Zlib-2.005  132 kB
[ebuild  N    ] dev-perl/Compress-Zlib-2.005  62 kB
[ebuild  N    ] dev-perl/IO-Zlib-1.07  10 kB
[ebuild  N    ] dev-perl/Archive-Tar-1.32  39 kB
[ebuild  N    ] dev-perl/module-build-0.28.08  USE="-test" 192 kB
[ebuild  N    ] dev-perl/ExtUtils-CBuilder-0.19  19 kB
[ebuild  N    ] dev-perl/extutils-parsexs-2.18  25 kB
[ebuild  N    ] dev-perl/Class-MethodMaker-2.10  88 kB
[ebuild  N    ] dev-perl/Shell-EnvImporter-1.04  15 kB
[ebuild  N    ] app-portage/autounmask-0.21  4 kB

Total: 21 packages (21 new), Size of downloads: 1,185 kB

Would you like to merge these packages? [Yes/No]
...
```

Im zweiten Schritt demaskieren wir einmal die neueste Apache-Version:

```
gentoo ~ # autounmask --pretend >=www-servers/apache-2.2.8

 autounmask version 0.21 (using PortageXS-0.02.07 and portage-2.1.3.19)

 * Using repository: /usr/portage

 * Using package.keywords file: /etc/portage/package.keywords
 * Using package.unmask file: /etc/portage/package.unmask

 * Unmasking www-servers/apache-2.2.8 and its dependencies.. this might
take a while..

 * Added '=www-servers/apache-2.2.8 ~x86' to /etc/portage/package.keywor
ds
 * Added '=app-admin/apache-tools-2.2.8 ~x86' to /etc/portage/package.ke
ywords
 * Restoring files because autounmask was called with the --pretend opti
on.
 * done!
```

Mit der Option `--pretend` zeigt das Tool zunächst, was es verändern würde. Dieser erste Schritt ist dringend zu empfehlen.

Der eigentliche Lauf, also ohne die `--pretend`-Option, trägt anschließend die Pakete inklusive Versionen in `/etc/portage/package.keywords` bzw. `/etc/portage/package.unmask` ein. Wer die Pakete generell freischalten möchte und nicht nur eine spezifische Paketversion, der kann die Option `--noversions` verwenden.

Kapitel 6

Die Datei /etc/make.conf

Die komplexen Konfigurationsvariablen `USE` und `ACCEPT_KEYWORDS`, die sich beide in der Datei `/etc/make.conf` definieren lassen, haben wir ausführlich behandelt. Schaut man sich die gut kommentierte Beispielkonfiguration in `/etc/make.conf.example` an, so stellen wir fest, dass damit ein Bruchteil der möglichen Konfigurationsvariablen abgedeckt ist. Allerdings haben die anderen Optionen deutlich geringere Auswirkungen.

Den Kopfteil, die Abschnitte `Build-time functionality`, `Host Setting`, `Host and optimization settings` und `Advanced Masking` aus der Datei `/etc/make.conf.example`, haben wir ebenfalls schon besprochen, und zwar als es im Abschnitt 1.6.1 ab Seite 38 um die Compiler-Flags (`CHOST`, `CFLAGS` und `CXXFLAGS`) im Zusammenhang mit der Installation ging.

Der zweite Teil der Datei `/etc/make.conf` konfiguriert Portage-Eigenschaften mit weniger zentraler Bedeutung für das System. Dazu gehören z. B. die verschiedenen Verzeichnisse, die für die Arbeit von Portage notwendig sind, die Konfiguration für die globalen Gentoo-Server und das Logging-Verhalten von Portage. Wer sich mit diesen exotischeren Konfigurationen

hier erst einmal nicht auseinander setzen möchte, springt zu Abschnitt 6.4 ab Seite 163 und konfiguriert nur kurz die wichtigsten Elemente.

Die Standard-Einstellungen für die Variablen, die wir im Folgenden beleuchten, werden diesmal nicht durch das Profil festgelegt, sondern finden sich in der Datei `/etc/make.globals` wieder, da diese Variablen nicht von der verwendeten Architektur abhängen.

6.1 Portage-Verzeichnisse

Im nächsten Abschnitt der Beispielkonfiguration (`Portage Directories`) finden sich einige Einstellungen zu zentralen Verzeichnissen von Portage.

```
# Portage Directories
# ===================
#
# Each of these settings controls an aspect of portage's storage and
# file system usage. If you change any of these, be sure it is available
# when you try to use portage. *** DO NOT INCLUDE A TRAILING "/" ***
#
# PORTAGE_TMPDIR is the location portage will use for compilations and
#     temporary storage of data. This can get VERY large depending upon
#     the application being installed.
#PORTAGE_TMPDIR=/var/tmp
#
# PORTDIR is the location of the portage tree. This is the repository
#     for all profile information as well as all ebuilds. If you change
#     this, you must update your /etc/make.profile symlink accordingly.
#PORTDIR=/usr/portage
#
# DISTDIR is where all of the source code tarballs will be placed for
#     emerges. The source code is maintained here unless you delete
#     it. The entire repository of tarballs for Gentoo is 9G. This is
#     considerably more than any user will ever download. 2-3G is
#     a large DISTDIR.
#DISTDIR=${PORTDIR}/distfiles
#
# PKGDIR is the location of binary packages that you can have created
#     with '--buildpkg' or '-b' while emerging a package. This can get
#     up to several hundred megs, or even a few gigs.
#PKGDIR=${PORTDIR}/packages
#
# PORT_LOGDIR is the location where portage will store all the logs it
#     creates from each individual merge. They are stored as
#     NNNN-$PF.log in the directory specified. This is disabled until
#     you enable it by providing a directory. Permissions will be
#     modified as needed IF the directory exists, otherwise logging will
#     be disabled. NNNN is the increment at the time the log is created.
#     Logs are thus sequential.
```

```
#PORT_LOGDIR=/var/log/portage
#
# PORTDIR_OVERLAY is a directory where local ebuilds may be stored
#     without concern that they will be deleted by rsync updates. Default
#     is not defined.
#PORTDIR_OVERLAY=/usr/local/portage
```

Das PORTDIR legt den Ort des zentralen Portage-Verzeichnisses fest. Im Normalfall ist dies /usr/portage.

Innerhalb dieses Verzeichnisses gibt es neben den verschiedenen Paket-Kategorien (siehe Kapitel 1.7.1) noch einige Dateien und Ordner mit besonderer Bedeutung. So legt Portage die Ordner distfiles und packages im Normalfall unterhalb von PORTDIR an. Beide können wir aber auch gesondert auslagern, da die dort abgelegten Dateien ein respektables Volumen auf der Festplatte einnehmen können. Den Zielort für heruntergeladene Quellpakete kann man über die Variable DISTDIR beeinflussen, den für binäre Pakete mit PKGDIR.

Diese Einstellungen zu verändern ist dann sinnvoll, wenn man die Dateien aus Platzgründen auslagern oder auf einem NFS-System bereithalten möchte. Letzteres kann besonders für das DISTDIR sinnvoll sein, wenn man in einem internen Netzwerk mehrere Maschinen mit Gentoo betreibt und vermeiden möchte, dass jede Maschine ihre eigenen Quellpakete herunterlädt. Innerhalb eines internen Netzwerkes mit einem Build-Host sollte man auch das PKGDIR vom zentralen Host aus an die Clients verteilen.

Die Standardeinstellung für das temporäre Portage-Verzeichnis legt die Variable PORTAGE_TMPDIR auf /var/tmp fest. Innerhalb dieses Verzeichnisses liegt dann normalerweise das Verzeichnis portage, in dem emerge alle Pakete zur Installation vorbereitet. Dieser Pfad lässt sich mit BUILD_PREFIX modifizieren. Die Standardeinstellung ist ${PORTAGE_TMPDIR}/portage.

PORT_LOGDIR definiert ein Verzeichnis, das den vollständigen Output aller Installationen und Updates enthält. Diese Transkripte erstellt emerge allerdings erst, sobald wir PORT_LOGDIR einen Wert zugewiesen haben. Das Verzeichnis kann ebenfalls eine nicht unerhebliche Größe annehmen, und man sollte es, so man es denn verwendet, gelegentlich leeren.

Schließlich legt PORTDIR_OVERLAY die Lage potentieller Overlays fest. Diese Variable schauen wir uns aber erst auf Seite 315 in Abschnitt 14 genauer an.

6.2 Mirrors

Um den Download von Daten und Paketen drehen sich die Einstellungen in den folgenden zwei Abschnitten der Datei make.conf.example: Fetching files und Synchronizing Portage.

```
# Fetching files
# ==============
#
# If you need to set a proxy for wget or lukemftp, add the appropriate
# "export ftp_proxy=<proxy>" and "export http_proxy=<proxy>" lines to
# /etc/profile if all users on your system should use them.
#
# Portage uses wget by default. Here are some settings for some
# alternate downloaders -- note that you need to merge these programs
# first before they will be available.
#
# Default fetch command (5 tries, passive ftp for firewall compatibility
)
#FETCHCOMMAND="/usr/bin/wget -t 5 -T 60 --passive-ftp ${URI} -P ${DIST
DIR}"
#RESUMECOMMAND="/usr/bin/wget -c -t 5 -T 60 --passive-ftp ${URI} -P ${
DISTDIR}"
#
# Using wget, ratelimiting downloads
#FETCHCOMMAND="/usr/bin/wget -t 5 -T 60 --passive-ftp --limit-rate=200k
${URI} -P ${DISTDIR}"
#RESUMECOMMAND="/usr/bin/wget -c -t 5 -T 60 --passive-ftp --limit-rate=2
00k ${URI} -P ${DISTDIR}"
#
# Lukemftp (BSD ftp):
#FETCHCOMMAND="/usr/bin/lukemftp -s -a -o ${DISTDIR}/${FILE} ${URI}"
#RESUMECOMMAND="/usr/bin/lukemftp -s -a -R -o ${DISTDIR}/${FILE} ${UR
I}"
#
# Portage uses GENTOO_MIRRORS to specify mirrors to use for source
# retrieval. The list is a space separated list which is read left to
# right. If you use another mirror we highly recommend leaving the
# default mirror at the end of the list so that portage will fall back
# to it if the files cannot be found on your specified mirror. We
# _HIGHLY_ recommend that you change this setting to a nearby mirror by
# merging and using the 'mirrorselect' tool.
#GENTOO_MIRRORS="<your_mirror_here> http://distfiles.gentoo.org http://w
ww.ibiblio.org/pub/Linux/distributions/gentoo"
#
# Portage uses PORTAGE_BINHOST to specify mirrors for prebuilt-binary
# packages. The list is a single entry specifying the full address of
# the directory serving the tbz2's for your system. Running emerge with
# either '--getbinpkg' or '--getbinpkgonly' will cause portage to
# retrieve the metadata from all packages in the directory specified,
# and use that data to determine what will be downloaded and merged.
# '-g' or '-gK' are the recommend parameters. Please consult the man
# pages and 'emerge --help' for more information. For FTP, the default
# connection is passive -- If you require an active connection, affix
# an asterisk (*) to the end of the host:port string before the path.
#PORTAGE_BINHOST="http://grp.mirror.site/gentoo/grp/1.4/i686/athlon-xp/"
# This ftp connection is passive ftp.
#PORTAGE_BINHOST="ftp://login:pass@grp.mirror.site/pub/grp/i686/athlon-x
```

```
p/"
# This ftp connection is active ftp.
#PORTAGE_BINHOST="ftp://login:pass@grp.mirror.site:21*/pub/grp/i686/athl
on-xp/"

# Synchronizing Portage
# =====================
#
# Each of these settings affects how Gentoo synchronizes your Portage
# tree. Synchronization is handled by rsync and these settings allow
# some control over how it is done.
#
# SYNC is the server used by rsync to retrieve a localized rsync mirror
#     rotation. This allows you to select servers that are
#     geographically close to you, yet still distribute the load over a
#     number of servers. Please do not single out specific rsync
#     mirrors. Doing so places undue stress on particular mirrors.
#     Instead you may use one of the following continent specific
#     rotations:
#
#   Default:       "rsync://rsync.gentoo.org/gentoo-portage"
#   North America: "rsync://rsync.namerica.gentoo.org/gentoo-portage"
#   South America: "rsync://rsync.samerica.gentoo.org/gentoo-portage"
#   Europe:        "rsync://rsync.europe.gentoo.org/gentoo-portage"
#   Asia:          "rsync://rsync.asia.gentoo.org/gentoo-portage"
#   Australia:     "rsync://rsync.au.gentoo.org/gentoo-portage"
#
#     If you have multiple Gentoo boxes, it is probably a good idea to
#     have only one of them sync from the rotations above. The other
#     boxes can then rsync from the local rsync server, reducing the
#     load on the mirrors. Instructions for setting up a local rsync
#     server are available here:
#     http://www.gentoo.org/doc/en/rsync.xml
#
#SYNC="rsync://rsync.gentoo.org/gentoo-portage"
#
# PORTAGE_RSYNC_RETRIES sets the number of times portage will attempt to
#     retrieve a current portage tree before it exits with an error.
#     This allows for a more successful retrieval without user
#     intervention most times.
#PORTAGE_RSYNC_RETRIES="3"
#
# PORTAGE_RSYNC_EXTRA_OPTS can be used to feed additional options to the
#     rsync command used by 'emerge --sync'. This will not change the
#     default options which are set by PORTAGE_RSYNC_OPTS (don't change
#     those unless you know  exactly what you're doing).
#PORTAGE_RSYNC_EXTRA_OPTS="
```

Portage benötigt für die Installation von Paketen zwei externe Datenquellen: Zum einen den *Portage-Baum*, der alle Paketdefinitionen enthält, zum anderen die eigentlichen Quellen der zu installierenden Programme. Mit

der Variablen SYNC legt man in der /etc/make.conf fest, von welchem Mirrorsystem man den Portage-Baum beziehen möchte, während man über GENTOO_MIRRORS die Mirror-Server für die Quellarchive bestimmt.

Der Portage-Baum liegt im Original in einem CVS-System, für das nur die Entwickler der Distribution Schreibrechte haben. Er ist über eine Vielzahl von rsync-Servern für die User zugänglich. Damit verteilt sich die Last der Zugriffe effizient, gibt es doch mehrere zentrale, kontinentspezifische rsync-Adressen, die die anfragenden Clients nach dem Zufallsprinzip auf die verfügbaren Server in der jeweiligen Region umleiten. Damit das funktioniert, muss die Variable SYNC entsprechend den Angaben in der Beispieldatei /etc/make.conf.example auf die korrekte Region gesetzt sein. Für Europa lautet die korrekte Adresse:

```
rsync://rsync.europe.gentoo.org/gentoo-portage
```

Alternativ kann man den passenden Server mit dem Tool mirrorselect auswählen. Es liegt als Paket mirrorselect in Kategorie app-portage:

```
gentoo ~ # emerge -av app-portage/mirrorselect

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-analyzer/netselect-0.3-r1  0 kB
[ebuild  N    ] app-portage/mirrorselect-1.2  0 kB

Total: 2 packages (2 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
```

Das Tool erlaubt es, den Mirror-Server direkt in die Datei /etc/make.conf hineinzuschreiben:

```
gentoo ~ # mirrorselect -i -r -o >> /etc/make.conf
gentoo ~ # cat /etc/make.conf
# These settings were set by the catalyst build script that automatically
# built this stage.
# Please consult /etc/make.conf.example for a more detailed example.
CFLAGS="-march=i686 -O2 -pipe"
CXXFLAGS="${CFLAGS}"
# This should not be changed unless you know exactly what you are doing.
# You should probably be using a different stage, instead.
CHOST="i686-pc-linux-gnu"
USE="apache2 mysql xml ldap -X"

SYNC="rsync://rsync.europe.gentoo.org/gentoo-portage"
```

-i (bzw. --interactive) startet den grafischen, interaktiven Modus; -r (bzw. --rsync) aktiviert die Rsync-Serverauswahl und -o (bzw. --output)

liefert als Ausgabe eine Konfigurationszeile, die wir sofort über >> an die Datei /etc/make.conf anhängen können.

Bleiben noch die eigentlichen Quellen der Programme: Portage ist zwar durchaus in der Lage, die Quelltextarchive der zu installierenden Software von den Original-Download-Locations (z. B. SourceForge) herunterzuladen. Um diese Server aber nicht zu stark zu strapazieren, spiegelt das Gentoo-Projekt alle Quellpakete auf eigenen Mirror-Servern. Auch hier vereinfacht mirrorselect die Auswahl:

```
gentoo ~ # mirrorselect -i -o
```

Hier fehlt das -r, um anzuzeigen, dass wir die Paketserver, nicht den Rsync-Server auswählen möchten.

Das Tool bietet auch die Möglichkeit, die Server im nicht-interaktiven Modus auf ihre Geschwindigkeit zu prüfen und die schnellsten Verbindungen auszuwählen. Diese Methode ist zu empfehlen, erleichtert sie die Auswahl doch wesentlich:

```
gentoo ~ # mirrorselect -o -s5 >> /etc/make.conf
* Downloading a list of mirrors... Got 199 mirrors.
* Stripping hosts that only support ipv6... Removed 8 of 199
* Using netselect to choose the top 5 mirrors...Done.
gentoo ~ # cat /etc/make.conf
# These settings were set by the catalyst build script that automatically
# built this stage.
# Please consult /etc/make.conf.example for a more detailed example.
CFLAGS="-march=i686 -O2 -pipe"
CXXFLAGS="${CFLAGS}"
# This should not be changed unless you know exactly what you are doing.
# You should probably be using a different stage, instead.
CHOST="i686-pc-linux-gnu"
USE="apache2 mysql xml ldap -X"

SYNC="rsync://rsync.europe.gentoo.org/gentoo-portage"

GENTOO_MIRRORS="ftp://pandemonium.tiscali.de/pub/gentoo/ http://gentoo.m
neisen.org/ ftp://sunsite.informatik.rwth-aachen.de/pub/Linux/gentoo ftp
://ftp.free.fr/mirrors/ftp.gentoo.org/ http://213.186.33.38/gentoo-distf
iles/"
```

Hier fällt -i weg, wodurch wir den interaktiven Modus abschalten. Außerdem selektiert -s5 (bzw. -servers 5) die fünf schnellsten Server und -o sorgt wieder für eine Ausgabe, die wir sofort an /etc/make.conf anhängen können. Eine detaillierte Beschreibung des Programms finden Sie im Kapitel 16.1.1 ab Seite 336.

Es gibt in der Datei /etc/make.conf einige Optionen, die den Download-Prozess beeinflussen. So lässt sich zum Beispiel das Kommando für den

Download über FETCHCOMMAND (bzw. RESUMECOMMAND für unterbrochene Downloads) modifizieren. Das ist z. B. dann sinnvoll, wenn man die Nutzung der Bandbreite optimieren und den Durchsatz limitieren möchte oder die Download-Menge so weit es geht reduzieren muss, weil man an einem schmalbandigen Internetzugang hängt. Die Möglichkeiten für solch einen Fall behandeln wir unter den Tipps und Tricks in Kapitel 16.1.2.

Für Rechner, auf denen wir auch binäre Pakete installieren möchten und die diese von einem zentralen Build-Host abholen, dient PORTAGE_BINHOST.

Für die Synchronisation des Portage-Baumes können wir über die Option PORTAGE_RSYNC_EXTRA_OPTS zusätzliche Parameter angeben, möglicherweise um z. B. Verzeichnisse mit `--exclude` auszuklammern, die man im Portage-Baum nicht überschrieben haben möchte. Angenommen man hätte Veränderungen im Verzeichnis `/usr/portage/www-apps` vorgenommen, die man bei der nächsten Synchronisation nicht überschreiben will, dann fügt man hier die Option `--exclude="/www-apps"` hinzu.

Im Normalfall sollte dieses Vorgehen allerdings absolut nicht notwendig sein. Veränderungen an den Ebuilds sollte man nie im eigentlichen Portage-Baum vornehmen, sondern in externen *Overlays* (Kapitel 14 ab Seite 309).

6.3 Fortgeschrittene Konfiguration

Kommen wir zu den Portage-Konfigurationsvariablen, die unter „Advanced features“ in `/etc/make.conf.example` gelistet werden. Nur wenige davon sind von zentraler Bedeutung, und wir beleuchten hier nicht alle.

```
# Advanced Features
# =================
#
# EMERGE_DEFAULT_OPTS allows emerge to act as if certain options are
#     specified on every run. Useful options include --ask, --verbose,
#     --usepkg and many others. Options that are not useful, such as
#     --help, are not filtered.
#EMERGE_DEFAULT_OPTS="
#
# INSTALL_MASK allows certain files to not be installed into your file
#     system. This is useful when you wish to filter out a certain set
#     of files from ever being installed, such as INSTALL.gz or TODO.gz
#INSTALL_MASK="
#
# MAKEOPTS provides extra options that may be passed to 'make' when a
#     program is compiled. Presently the only use is for specifying
#     the number of parallel makes (-j) to perform. The suggested
#     number for parallel makes is CPUs+1.
```

```
#MAKEOPTS="-j2"
#
# PORTAGE_NICENESS provides a default increment to emerge's niceness
#     level. Note: This is an increment. Running emerge in a niced
#     environment will reduce it further. Default is unset.
#PORTAGE_NICENESS=3
#
# AUTOCLEAN enables portage to automatically clean out older or
#     overlapping packages from the system after every successful merge.
#     This is the same as running 'emerge -c' after every merge. Set
#     with: "yes" or "no". This does not affect the unpacked source. See
#     'noclean' below.
#
#     Warning: AUTOCLEAN="no" can cause serious problems due to
#              overlapping packages.  Do not use it unless absolutely
#              necessary!
#AUTOCLEAN="yes"
#
# PORTAGE_TMPFS is a location where portage may create temporary files.
#     If specified, portage will use this directory whenever possible
#     for all rapid operations such as lockfiles and transient data.
#     It is _highly_ recommended that this be a tmpfs or ramdisk. Do not
#     set this to anything that does not give a significant performance
#     enhancement and proper FS compliance for locks and read/write.
#     /dev/shm is a glibc mandated tmpfs, and should be a reasonable
#     setting for all linux kernel+glibc based systems.
#PORTAGE_TMPFS="/dev/shm"
```

6.3.1 Make-Optionen

Die Variable `MAKEOPTS` beeinflusst ebenso wie die `CFLAGS`-Einstellungen nahezu jeden Installationsvorgang unter Gentoo. Dadurch können wir jedem `make`-Aufruf zusätzliche Kommandozeilen-Parameter übergeben. Entsprechend vorsichtig sollte man mit dieser Einstellung sein. Im Normalfall legt man hier nur mit der Option `--jobs` (bzw. `-j`) fest, wie viele Operation `make` gleichzeitig starten soll. Als Faustregel gilt die Anzahl der CPUs im Rechner plus eins.

6.3.2 Portage-Rechenzeit

Die Einstellung `PORTAGE_NICENESS` erlaubt die Priorität des `emerge`-Prozesses automatisch zu reduzieren und damit anderen Prozessen im Vordergrund mehr Rechenzeit zuzugestehen. Dies ist vor allem bei Desktop-Systemen sinnvoll, wenn ein im Hintergrund laufender `emerge`-Prozess das System in seinen Reaktionen merklich aus bremst.

6.3.3 Features

Kommen wir zu einem längeren Abschnitt der fortgeschrittenen Konfiguration, der Variable FEATURES:

```
# FEATURES are settings that affect the functionality of portage. Most
#     of these settings are for developer use, but some are available
#     to non- developers as well.
#
#  'assume-digests'
#                when committing work to cvs with repoman(1), assume
#                that all existing SRC_URI digests are correct.  This
#                feature also affects digest generation via ebuild(1)
#                and emerge(1) (emerge generates digests only when the
#                'digest' feature is enabled).
#  'buildpkg'    causes binary packages to be created of all packages
#                that  are being merged.
#  'ccache'      enable support for the dev-util/ccache package, which
#                can noticably decrease the time needed to remerge
#                previously built packages.
#  'confcache'   enable confcache support; speeds up autotool based
#                configure calls
#  'collision-protect'
#                prevents packages from overwriting files that are owned
#                by another package or by no package at all.
#  'cvs'         causes portage to enable all cvs features (commits,
#                adds), and to apply all USE flags in SRC_URI for
#                digests -- for developers only.
#  'digest'      autogenerate digests for packages when running the
#                emerge(1) command.  If the 'assume-digests' feature is
#                also enabled then existing SRC_URI digests will be
#                reused whenever they are available.
#  'distcc'      enables distcc support via CC.
#  'distlocks'   enables distfiles locking using fcntl or hardlinks.
#                This is enabled by default. Tools exist to help clean
#                the locks after crashes: /usr/lib/portage/bin/clean_loc
ks.
#  'fixpackages' allows portage to fix binary packages that are stored
#                in PKGDIR. This can consume a lot of time.
#                'fixpackages' is also a script that can be run at any
#                given time to force the same actions.
#  'gpg'         enables basic verification of Manifest files using gpg.
#                This features is UNDER DEVELOPMENT and reacts to
#                features of strict and severe. Heavy use of gpg sigs is
#                coming.
#  'keeptemp'    prevents the clean phase from deleting the temp files
#                ($T) from a merge.
#  'keepwork'    prevents the clean phase from deleting the WORKDIR.
#  'test'        causes ebuilds to perform testing phases if they are
#                capable of it. Some packages support this automatically
#                 via makefiles.
#  'metadata-transfer'
```

```
#               automatically perform a metadata transfer when 'emerge
#               --sync' is run.
#  'noauto'     causes ebuild to perform only the action requested and
#               not any other required actions like clean or unpack --
#               for debugging purposes only.
#  'noclean'    prevents portage from removing the source and temporary
#               files  after a merge -- for debugging purposes only.
#  'nostrip'    prevents the stripping of binaries.
#  'notitles'   disables xterm titlebar updates (which contain status
#               info).
#  'parallel-fetch'
#               do fetching in parallel to compilation
#  'sandbox'    enables sandboxing when running emerge and ebuild.
#  'strict'     causes portage to react strongly to conditions that are
#               potentially dangerous, like missing/incorrect Manifest
#               files.
#  'stricter'   causes portage to react strongly to conditions that may
#               conflict with system security provisions (for example
#               textrels, executable stacks).
#  'userfetch'  when portage is run as root, drop privileges to
#               portage:portage during the fetching of package sources.
#  'userpriv'   allows portage to drop root privileges while it is
#               compiling, as a security measure.  As a side effect
#               this can remove  sandbox access violations for users.
#  'usersandbox' enables sandboxing while portage is running under
#               userpriv.
#FEATURES="sandbox buildpkg ccache distcc userpriv usersandbox notitles
 noclean noauto cvs keeptemp keepwork"
#FEATURES="sandbox ccache distcc distlocks"
```

Über die FEATURES-Einstellung kann man eine Vielzahl verschiedener Eigenschaften von Portage zusätzlich aktivieren. Einige davon sind nur für Entwickler interessant, manche aber auch für die Nutzer. Die Aktivierung erfolgt, indem wir die entsprechenden Schlagworte zur FEATURES-Variable hinzufügen. Wollen wir bestimmte Eigenschaften deaktivieren, dann müssen wir den entsprechenden Flags, wie bei den USE-Flags auch, ein Minus voranstellen. Hier ein Beispiel, welches das Erstellen binärer Pakete aktiviert und das Sicherheitsfeature `sandbox` deaktiviert:

```
FEATURES="buildpkg -sandbox"
```

Es folgt eine Übersicht über die wichtigsten Eigenschaften:

`assume-digests`
: Ein spezielles Feature für Gentoo-Entwickler. Wenn es aktiviert ist, beschwert sich `repoman`, ein Werkzeug zur Qualitätskontrolle, nicht mehr über nicht verifizierbare Prüfsummen, wenn einzelne Quellpakete fehlen, um die Prüfsummen zu generieren. Diese Eigenschaft ist nur für Entwickler interessant und sollte nicht aktiviert sein.

`buildpkg`
: Veranlasst Portage dazu, bei jedem Installationsvorgang ein binäres Paket zu erstellen und unter `/usr/portage/packages` (bzw. `PKGDIR`, s. o.) abzulegen. Die Option sollte man nur aktivieren, wenn man wirklich beabsichtigt, einen Buildhost bereitzustellen.

`buildsyspkg`
: Hat den gleichen Effekt wie `buildpkg`, wirkt sich allerdings nur auf System-Pakete aus (siehe Seite 219).

`ccache`
: Diese Option aktiviert einen Compiler-Cache und kann das Kompilieren unter bestimmten Bedingungen beschleunigen. Dieses Feature erfordert einige zusätzliche Optionen, die wir genauer ab Seite 364 beschreiben. Ohne diese zusätzlichen Konfigurationen ist das Aktivieren dieser Option sinnlos.

`confcache`
: Sollte ähnlich wie `ccache` die Ergebnisse des `configure`-Laufes in einem Cache zu speichern und so bei wiederkehrenden `configure`-Aufgaben Zeit zu sparen. In neueren Portage-Versionen wurde diese Option aufgrund von Problemen enfernt und sie sollte nicht verwendet werden.

`collision-protect`
: Es gibt seltene Fälle, in denen zwei Pakete eine Datei an die gleiche Stelle des Dateibaumes schreiben. Gelegentlich legt man auch als Benutzer manuell Dateien an, die im Konflikt mit den Dateien eines zu installierenden Pakets stehen. Standardmäßig überschreibt Portage schon existierende Dateien im Dateisystem einfach. Die Ausnahme sind geschützte Konfigurationsdateien, aber diesen Mechanismus erläutern wir erst im Kapitel 10 genauer.

  Im Normalfall sollte man als Benutzer auch keine Dateien innerhalb der System-Verzeichnisse anlegen und das Standardverhalten von Portage, im Zweifelsfall vorhandene Dateien zu überschreiben, entspricht dieser Voraussetzung.

  Wer allerdings gerne auch außerhalb des Paketmanagement-Systems Veränderungen an Systemdateien vornimmt, der fährt sicherer, indem er `collision-protect` in die `FEATURES`-Variable mit aufnimmt. Portage weigert sich dann, die schon vorhandene Datei zu überschreiben, und bricht die Installation im Notfall mit folgender Meldung ab:

```
* checking X files for package collisions
* This package is blocked because it wants to overwrite
* files belonging to other packages (see list below).
* If you have no clue what this is all about report it
```

```
* as a bug for this package on http://bugs.gentoo.org

package app-collide/test-1.0 NOT merged

Detected file collision(s):

     '/usr/share/man/man1/test.1.bz2'

Searching all installed packages for file collisions...
Press Ctrl-C to Stop

None of the installed packages claim the above file(s).
```

`cvs`
: Aktiviert den Entwicklermodus für Portage. Wir sollten diese Option nicht aktivieren.

`digest`
: Automatisches Erstellen von Prüfsummen, wenn man `emerge` aufruft, um eine Software zu installieren. Auch dies ist ein reines Feature für Entwickler, das man definitiv *nicht* aktivieren sollte. Es schaltet die Überprüfung der Prüfsummen aus und reduziert damit die Sicherheit des Systems.

`distcc`
: Diese Option aktiviert die Unterstützung für verteiltes Kompilieren. Der Aufbau eines solchen Netzwerks kann den Kompiliervorgang vor allem bei großen Paketen oder für langsame Rechner deutlich beschleunigen, aber erfordert auch einen gewissen Konfigurationsaufwand, den wir ab Seite 367 genauer beleuchten.

`distlocks`
: Dieses Feature ist standardmäßig aktiviert und erlaubt Portage, den Zugriff auf die Quellpakete während des Installationsvorgangs zu sperren. Diese Option sollte aktiviert bleiben.

`fixpackages`
: Das Aktivieren führt dazu, dass Portage die binären Pakete nach der Portage-Synchronisation abgleicht. Da im Normalfall nur Buildhosts binäre Pakete erstellen und die Aktion viel Zeit in Anspruch nehmen kann, ist diese Eigenschaft standardmäßig deaktiviert und sollte es im Normalfall auch bleiben.

`getbinpkg`
: Portage wird versuchen, ausschließlich binäre Pakete von einem Buildhost zu installieren.

`gpg`
: Aktiviert die Unterstützung für signierte Ebuilds. Das erhöht zwar die Sicherheit des Systems (theoretisch), wird aber derzeit praktisch noch nicht genutzt und ist von daher nicht zu empfehlen.

`keeptemp` und `keepwork`
: Während der Installation eines Pakets und vor allem beim Kompilieren entstehen im `$PORTAGE_TMPDIR` (siehe Seite 143) eine Reihe temporärer Dateien (siehe auch die Eigenschaft `sandbox` weiter unten), die `emerge` nur für die Installation braucht. Portage löscht diese im Normalfall nach der Installation. Wer zu viel Speicherplatz hat oder wissen will, was Portage während der Installation treibt, kann diese Eigenschaften aktivieren. Dir meisten Dateien entstehen im Arbeitsverzeichnis (`$PORTAGE_TMPDIR/portage/`*KATEGORIE*`/`*PAKET*`/work`) und wir können es mit `keepwork` vor dem Löschen bewahren.

`metadata-transfer`
: Diese Eigenschaft ist standardmäßig aktiviert und führt dazu, dass jedes `emerge --sync` automatisch `emerge --metadata` nach sich zieht. Damit aktualisiert `emerge` nach dem Synchronisieren den Metadaten-Cache. Dieser Cache ist notwendig, um Portage beim Zugriff auf die Ebuild-Daten zu beschleunigen. Es gibt eigentlich keinen Grund, diese Option zu deaktivieren.

`noauto`
: Verhindert einige Portage-Aktionen, die normalerweise automatisch mit einer Installation verknüpft sind (wie z. B. das automatische Entfernen alter Versionen). Diese Option sollte man nicht aktivieren.

`nodoc,noman,noinfo`
: Diese drei Optionen unterdrücken die Installation der Hilfe-Seiten in `/usr/share`. `nodoc` verhindert die Installation in `/usr/share/doc`, `noinfo` in `/usr/share/info` und `noman` in `/usr/share/man`. Da Dokumentation generell etwas eher Nützliches ist, sollten diese Optionen selten Gebrauch finden.

`nostrip`
: Im Normalfall entfernt Portage nach dem Kompilieren der Pakete alle Debugging-Symbole aus den erstellten Binaries und verkleinert die Programm-Dateien damit merklich. Möchte man eine Software debuggen, ist dieses Verhalten nicht erwünscht. Innerhalb der Datei `/etc/make.conf` sollte man diese Eigenschaft normalerweise nicht setzen, da sie den Platzbedarf des Systems merklich vergrößert. Aber es ist sinnvoll, sie bei Bedarf als Umgebungsvariable zu setzten (s. u.).

`notitles`
: Während `emerge` läuft, versucht das Programm, die Titelleiste des

umgebenden X-Terminals zu modifizieren und anzuzeigen, welches Paket es gerade bearbeitet. Dieses Verhalten kann man über diese Option deaktivieren.

`parallel-fetch`
: Im Normalfall arbeitet Portage bei der Installation mehrerer Pakete einen Schritt nach dem anderen ab. Prinzipiell spricht aber nichts dagegen, während des Kompiliervorgangs des einen Pakets schon den Source-Code für das nächste Paket herunterzuladen, da die Benutzung des langsamen Netzwerks die CPU verhältnismäßig wenig beansprucht. `parallel-fetch` erlaubt `emerge`, den Download-Vorgang für ein zweites Paket in die Kompilierphase des vorangehenden Paketes zu ziehen.

`sandbox`
: Während des Kompilierens und der Installation eines Pakets durchläuft `emerge` verschiedene Phasen, von denen die meisten innerhalb von `/var/tmp/portage` (bzw. `PORTAGE_TMPDIR`) stattfinden. Erst wenn `emerge` die Software fertig erstellt hat und die *Merge*-Phase erreicht ist, beginnt Portage, die Arbeit aus dem temporären Verzeichnis in das eigentliche System zu übertragen. In allen vorangehenden Phasen sollten keinerlei Aktionen irgendeine Auswirkung auf Dateien des Systems außerhalb des entsprechenden temporären Verzeichnisses haben. Als zusätzliche Sicherheit dient Portage die Funktionalität der *Sandbox* (Sandkasten).

  Das entsprechende Tool erlaubt ausschließlich Zugriffe am temporären Arbeitsort und führt zum sofortigen Abbruch des `emerge`-Prozesses, wenn es einen Zugriff auf das umliegende, geschützte Dateisystem feststellt. Es sollte keinen Grund geben, diese Portage-Eigenschaft zu deaktivieren. Die Sandbox funktioniert nicht, wenn wir die Option `userpriv` in den Features setzen. In diesem Fall müssen wir alternativ die Option `usersandbox` zu den `FEATURES` hinzufügen.

`sfperms`
: Hier müssen wir kurz ausholen, um die genaue Funktionsweise dieser Option zu veranschaulichen: Unix stattet ein Programm im Normalfall mit den Rechten des aufrufenden Benutzers aus. Es gibt nur wenige Programme, die speziell als *Set User ID* bzw. *SUID* markiert sind. Diese laufen beim Aufruf nicht mit den Rechten des Aufrufenden, sondern mit den Rechten des Besitzers der Datei. Vielfach gehört die Programmdatei dem `root`-Benutzer. Ein Programm, das automatisch mit `root`-Rechten läuft, stellt auch immer ein Sicherheitsrisiko dar. Somit muss die Funktionsweise der Programme extrem streng kontrolliert sein, damit kein Benutzer sich über die Verwendung solcher Programme erhöhte Rechte verschaffen kann. Ganz verzichten kann man auf diesen besonderen Dateityp allerdings nicht.

Im Normalfall reicht es, wenn ein Benutzer ein SUID-Programm ausführen kann. Die Datei selbst muss dabei für den Benutzer nicht zwingend lesbar sein. Trotzdem installieren viele Pakete ein SUID-Programm auch lesbar für alle Benutzer des Systems. Die Option `sfperms` korrigiert diese Situation automatisch. Es schadet nicht, die Option mit in `FEATURES` aufzunehmen. Der Sicherheitsgewinn ist andererseits auch nicht allzu groß.

`strict`
: Diese Eigenschaft bringt Portage dazu, auch bei kleineren Ebuild-Fehlern sofort abzubrechen. Portage bricht so z. B. die Installation ab, falls die Checksummen der Dateien des Ebuilds oder des Source-Code nicht mit den im Ebuild angegebenen Werten übereinstimmen. Die hier auftretenden Fehler werden durch Unachtsamkeiten der Gentoo-Entwickler verursacht. Wenn Portage die Installation aufgrund solcher Fehler abbricht, wird der Benutzer aufgefordert, den Fehler in der Gentoo-Bug-Datenbank zu melden. Diese Eigenschaft sollte ebenfalls aktiviert sein, da sie die Sicherheit des Systems erhöht.

`stricter`
: Macht Portage nochmals strikter als mit `strict`. Diese Eigenschaft sollten wir nicht aktivieren, da `emerge` andernfalls einige Pakete nicht installieren kann. Die Fehler, aufgrund derer Portage hier abbrechen würde, sind zwar auch Fehler der Pakete, aber manche dieser Probleme sind durch die Entwickler des Quellcodes verursacht. Oft wäre der Aufwand zu hoch, diese Fehler innerhalb der Paketdefinition zu korrigieren. Deshalb sind nicht alle Pakete im Portage-Baum absolut fehlerfrei, wenn man das `stricter`-Kriterium heranzieht.

`suidctl`
: Auf den meisten Systemen sollten möglichst wenige Programme beim Ausführen die Rechte des Programmbesitzers übernehmen. Setzt man `suidctl` in der `FEATURES`-Variable, dann ist es möglich, in der Datei `/etc/portage/suidctl.conf` explizit festzulegen, welche Dateien die SUID-Eigenschaft besitzen dürfen. Versucht ein Paket, eine Datei, die *nicht* in `/etc/portage/suidctl.conf` gelistet ist, mit SUID-Bit zu installieren, entfernt `emerge` dieses automatisch und installiert die Datei ohne SUID-Eigenschaft.

`test`
: Über dieses Schlagwort kann man eventuelle Test-Suites der Ebuilds aktivieren. Viele Software-Pakete bringen heutzutage automatisierte Test-Systeme mit. Das Durchlaufen der Tests verzögert auf der anderen Seite den Vorgang des Kompilierens.

`userfetch`
: Veranlasst `emerge`, die Root-Rechte abzugeben, bevor es Dateien herunterlädt. `emerge` agiert dann als `portage`-Benutzer. Sinnvolle Option, da die Sicherheit erhöht wird.

`userpriv`
: Veranlasst Portage, die Root-Rechte abzugeben, bevor `emerge` ein Paket kompiliert. `emerge` agiert dann als `portage`-Benutzer. Sinnvolle Option, da die Sicherheit erhöht wird.

`usersandbox`
: `userpriv` ist inkompatibel zu der `sandbox`-Eigenschaft. Will man die Sandbox auch für Portage unter `userpriv` aktivieren, so muss man die Eigenschaft `usersandbox` angeben.

Eine sinnvolle Kombination für die Variable `FEATURES` wäre z. B.:

```
FEATURES="sandbox parallel-fetch strict distlocks"
```

Übrigens können wir Features genau wie USE-Flags kurzfristig für einen `emerge`-Aufruf aktivieren, indem wir dem `emerge`-Befehl die `FEATURES`-Variable als Umgebungsvariable voran stellen. Dies macht z. B. für das Aktivieren der Paket-eigenen Tests Sinn, da man nicht bei jedem Paket die teilweise sehr zeitaufwendigen Test-Verfahren durchlaufen möchte. Folgender Befehl würde einmalig für die Installation von `app-admin/webapp-config` das Durchlaufen der Test-Suite aktivieren:

```
gentoo ~ # FEATURES="test" emerge -av app-admin/webapp-config

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-admin/webapp-config-1.50.15  95 kB

Total: 1 package (1 new), Size of downloads: 95 kB

Would you like to merge these packages? [Yes/No] No
```

Wenn Sie möchten, können sie das Paket hier installieren. Wir fügen es unserem System spätestens in Kapitel 12.4 hinzu.

6.3.4 Logging

Am Ende der Paketinstallation zeigt `emerge` häufig abschließende Informationen, die oft zusätzliche Konfigurations- oder Updatehinweise enthalten. Wie Portage mit diesen umgeht, lässt sich im letzten Abschnitt der `make.conf`-Datei beeinflussen:

```
# logging related variables:
# PORTAGE_ELOG_CLASSES: selects messages to be logged, possible values
#                       are:
#                          info, warn, error, log, qa
#                       Warning: commenting this will disable elog
PORTAGE_ELOG_CLASSES="warn error log"

# PORTAGE_ELOG_SYSTEM: selects the module(s) to process the log
#                      messages. Modules included in portage are (empty
#                      means logging is disabled):
#                          save (saves one log per package in
#                                $PORT_LOGDIR/elog,
#                                /var/log/portage/elog if $PORT_LOGDIR
#                                is unset)
#                          custom (passes all messages to
#                                  $PORTAGE_ELOG_COMMAND)
#                          syslog (sends all messages to syslog)
#                          mail (send all messages to the mailserver
#                                defined in $PORTAGE_ELOG_MAILURI)
#                          save_summary (like "save" but merges all
#                                        messages in
#                                        $PORT_LOGDIR/elog/summary.log,
#                                        /var/log/portage/elog/summary.l
og
#                                        if $PORT_LOGDIR is unset)
#                          mail_summary (like "mail" but sends all
#                                        messages in a single mail when
#                                        emerge exits)
#                      To use elog you should enable at least one module
#PORTAGE_ELOG_SYSTEM="save mail"

# PORTAGE_ELOG_COMMAND: only used with the "custom" logging module.
#                       Specifies a command to process log messages. Two
#                       variables are expanded:
#                          ${PACKAGE} - expands to the cpv entry of the
#                                       processed package (see $PVR in
#                                       ebuild(5))
#                          ${LOGFILE} - absolute path to the logfile
#Both variables have to be quoted with single quotes
#PORTAGE_ELOG_COMMAND="/path/to/logprocessor -p '\${PACKAGE}' -f '
\${LOGFILE}'"

# PORTAGE_ELOG_MAILURI: this variable holds all important settings for
#                       the mail module. In most cases listing the
#                       recipient address and the receiving mailserver
#                       should be sufficient, but you can also use
#                       advanced settings like authentication or TLS.
#                       The full syntax is:
#                           address [[user:passwd@]mailserver[:port]]
#                       where
#                           address:    recipient address
#                           user:       username for smtp auth (defaults
```

```
#                                       to none)
#                             passwd:   password for smtp auth (defaults
#                                       to none)
#                             mailserver: smtp server that should be used
#                                       to deliver the mail (defaults to
#                                       localhost)
#                                       alternatively this can also be a
#                                       the path to a sendmail binary if
#                                       you don't want to use smtp
#                             port:     port to use on the given smtp
#                                       server (defaults to 25, values >
#                                       100000 indicate that starttls
#                                       should be used on (port-100000))
#                         Examples:
#PORTAGE_ELOG_MAILURI="root@localhost localhost" (this is also the defau
lt setting)
#PORTAGE_ELOG_MAILURI="user@some.domain mail.some.domain" (sends mails t
o user@some.domain using the mailserver mail.some.domain)
#PORTAGE_ELOG_MAILURI="user@some.domain user:secret@mail.some.domain:100
465" (this is left uncommented as a reader exercise ;)

# PORTAGE_ELOG_MAILFROM: you can set the from-address of logmails with
#                        this variable, if unset mails are sent by
#                        "portage" (this default may fail in some
#                        environments).
#PORTAGE_ELOG_MAILFROM="portage@some.domain"

# PORTAGE_ELOG_MAILSUBJECT: template string to be used as subject for
#                           logmails. The following variables are
#                           expanded:
#                               ${PACKAGE} - see description of
#                                            PORTAGE_ELOG_COMMAND
#                               ${HOST} - FQDN of the host portage is
#                                         running on
#PORTAGE_ELOG_MAILSUBJECT="package \${PACKAGE} merged on \${HOST
} with notice"
```

Ein Problem, mit dem sich `emerge` lange herumgeschlagen hat, war die Tatsache, dass es diese Konfigurations- bzw. Update-Hinweise bei der Installation mehrerer Pakete in Reihe nur jeweils nach Abschluss eines Paketes angezeigt hat. Wenn die endlosen Zeilen des Kompilierens am Auge vorbeirauschen, dann gehen diese kurzen Zeilen wichtigen Outputs leicht unter. Es besteht zwar die Möglichkeit, die auf Seite 143 erwähnte Variable `PORT_LOGDIR` zu aktivieren und damit eine Ausgabe aller `emerge`-Prozesse in dem angegebenen Verzeichnis zu erhalten. Allerdings ist auch diese Ausgabe recht unübersichtlich, da `emerge` hier alle Meldungen von `configure`, `make`, etc. sammelt.

Neueren Portage-Versionen gelingt es aber, die Hinweise gesammelt am Ende der Installation auszugeben. Außerdem bieten sie ein spezielles *Logging-Framework*, das als `elog` bezeichnet wird und das es dem Nutzer erlaubt,

mit den wichtigen Ausgabezeilen des `emerge`-Prozesses beliebig flexibel umzugehen.

Um die Log-Messages abzufangen, müssen wir in einem ersten Schritt festlegen, welche Informationen wir erhalten möchten. Dazu können wir verschiedene Log-Stufen in die Variable `PORTAGE_ELOG_CLASSES` eintragen. Es ist zu empfehlen, hier, wie auch standardmäßig vorgegeben, die Stufen `log`, `warn` und `error` anzugeben:

```
PORTAGE_ELOG_CLASSES="warn error log"
```

Wer mag, kann auch `info` hinzufügen, da viele Ebuilds auch hier relevante Informationen ausgeben. Allerdings ist der Anteil unnützer Mitteilungen schon deutlich höher als bei den anderen drei Klassen und stört das „Lesevergnügen“.

Erst im zweiten Schritt veranlassen wir das System dazu, wirklich Informationen zu sammeln, indem wir über `PORTAGE_ELOG_SYSTEM` die verarbeitenden Module wählen. Als Minimum sollte man hier `save_summary` eintragen, damit `emerge` die Nachrichten in der Datei `$PORT_LOGDIR/elog/summary.log` speichert. Bei neueren Portage-Versionen ist dies die Standardeinstellung.

```
PORTAGE_ELOG_SYSTEM="save_summary"
```

Damit sollte das `elog`-System seine Arbeit aufnehmen. Unter der Annahme, dass `$PORT_LOGDIR` auf `/var/log/portage` verweist, sollten sich dort nach einer erneuten Installation von Portage entsprechende Log-Dateien befinden:

```
gentoo ~ # emerge -av sys-apps/portage

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] sys-apps/portage-2.1.2.2  USE="-build -doc -epydoc (-sel
inux)" LINGUAS="-pl" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
gentoo ~ # ls /var/log/portage/elog/
summary.log
gentoo ~ # cat /var/log/portage/elog/summary.log
>>> Messages generated by process 8161 on 2007-09-13 12:53:59 for
package sys-apps/portage-2.1.2.2:

LOG: postinst
```

```
See NEWS and RELEASE-NOTES for further changes.

For help with using portage please consult the Gentoo Handbook
at http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=3


WARN: postinst
In portage-2.1.2, installation actions do not necessarily pull in build
time
dependencies that are not strictly required.  This behavior is adjustabl
e
via the new --with-bdeps option that is documented in the emerge(1) man
page.
For more information regarding this change, please refer to bug #148870.
```

`/var/log/portage/elog/summary.log` sammelt jetzt den Output jeder einzelnen `emerge` Operation, die entsprechend dem Paketnamen benannt und mit einem Zeitstempel versehen ist. Innerhalb der Log-Datei sortiert `emerge` die Meldungen nach Abfolge und Typ der Nachricht. Im angegebenen Beispielfall informiert vor allem die letztere Nachricht über ein geändertes Verhalten bei neueren Portage-Versionen.

Auch wenn Log-Dateien nützlich sind, lesen wir sie doch meist erst in Situationen, in denen irgendetwas nicht funktioniert. Bei vielen der von Portage ausgegebenen Instruktionen handelt es sich jedoch um aktive Handlungsanweisungen, um eine Konfiguration abzuschließen oder die Daten eines Paketes zu aktualisieren. Daher ist es sinnvoll, diese Nachrichten als E-Mail an den Nutzer zu übermitteln. So erfahren die Informationen die ihnen gebührende Aufmerksamkeit und wir können die Mail nach der Bearbeitung löschen.

Dieses Verhalten unterstützt `emerge` ebenfalls. Um die E-Mail-Benachrichtigung zu aktivieren, muss das `mail`-Modul zu `PORTAGE_ELOG_SYSTEM` hinzugefügt werden:

```
PORTAGE_ELOG_SYSTEM="save_summary mail"
```

Wer mag, kann an dieser Stelle auch `mail_summary` angeben und erhält dann eine kombinierte Mail für jeden `emerge`-Lauf anstatt eine Mail pro installiertem Paket.

Zusätzlich müssen wir die Zieladresse und den Mailserver angeben. Dies geschieht in der Variable `PORTAGE_ELOG_MAILURI`.

```
PORTAGE_ELOG_MAILURI="ich@example.com mail.example.com"
```

Als weitere Zielorte kann `emerge` die Informationen auch über `syslog` ausgeben, indem man das Modul `syslog` in `PORTAGE_ELOG_SYSTEM` angibt.

Und als letzte Alternative – für Bastler – kann `emerge` die Log-Einträge durch eine eigene Pipeline schicken. Dazu muss das `custom`-Modul zu `PORTAGE_ELOG_SYSTEM` hinzugefügt werden. Den auszuführenden Befehl legt man in in `PORTAGE_ELOG_COMMAND` fest.

6.3.5 Exotischere Konfigurationsvariablen

Kommen wir noch zu einigen eher selten verwendeten und vielleicht exotischeren Konfigurationsvariablen, die sich teilweise auch in der Beispieldatei nicht wiederfinden.

`EMERGE_DEFAULT_OPTS`
: Optionen, die wir bei *jedem* `emerge`-Aufruf anhängen wollen. Diese Möglichkeit eignet sich nur für Optionen wie `--verbose` (siehe Seite 96) oder `--ask` (siehe Seite 100).

`CLEAN_DELAY`
: Bevor `emerge` ein Paket wirklich deinstalliert, pausiert das Programm kurz, um dem Benutzer die Chance zu geben, doch noch abzubrechen. Die Länge der Pause in Sekunden legt `CLEAN_DELAY` fest.

`COLLISION_IGNORE`
: Haben wir in der Variable `FEATURES` die Option `collision-protect` (siehe Seite 152) gesetzt, dann können wir in `COLLISION_IGNORE` einzelne Dateien festlegen, die `emerge` beim Kollisions-Check nicht beachtet soll. Diese kann `emerge` also überschreiben.

`EMERGE_WARNING_DELAY`
: In kritischen Situationen warnt Portage den Benutzer und wartet die hier angegebene Zahl an Sekunden, bevor `emerge` eine möglicherweise kritische Aktion fortführt.

`EBEEP_IGNORE`
: Ignoriert den `ebeep`-Befehl, den die Entwickler in manchen Ebuilds verwenden, um den Benutzer auf einen wichtigen Hinweis aufmerksam zu machen. Manche empfinden diesen Ton als sehr störend; mit dieser Option lässt er sich abschalten.

`EPAUSE_IGNORE`
: Analog zum `ebeep`-Befehl enthalten manche Ebuilds den `epause`-Befehl, der zu einer kurzen Pause führt und damit auf einen wichtigen Hinweis aufmerksam macht. Dieses Verhalten lässt sich mit `EPAUSE_IGNORE="yes"` unterdrücken.

`INSTALL_MASK`
: Hier können wir Verzeichnisse angeben, in denen Portage nichts installieren soll. Möchte ein Paket Dateien in einem gelisteten Ordner

installieren, ignoriert `emerge` diese. Die Option ist mit Vorsicht zu genießen, denn die meisten Dateien eines Paketes installiert Portage schließlich nicht ohne Grund.

`PORTAGE_COMPRESS` und `PORTAGE_COMPRESS_FLAGS`
: `PORTAGE_COMPRESS` legt das Programm fest, mit dem `emerge` Dokumentationsdateien bei der Installation komprimiert. Im Normalfall ist dies `bzip2`. `PORTAGE_COMPRESS_FLAGS` legt eventuelle Optionen fest. Standardmäßig enthält diese Variable den Wert `-9` für den höchsten Kompressionsgrad.

6.4 Fazit

Die Konfigurations-Datei `make.conf` bietet zweifelsohne eine extreme Menge möglicher Optionen, die vor allem den Anfänger leicht überfordern können. Eine sinnvolle Möglichkeit ist es, die Datei `make.conf` bis auf die während der Installation vorgenommenen Eintragungen unverändert zu lassen:

```
gentoo ~ # cat /etc/make.conf
CFLAGS="-O2 -march=i686 -pipe"
CXXFLAGS="${CFLAGS}"
CHOST="i686-pc-linux-gnu"
USE="apache2 ldap mysql xml -X"
```

Dieses Vorgehen hat, abgesehen davon, dass man sich die Zeit spart, sich über selten verwendete Eigenschaften von `emerge` den Kopf zu zerbrechen, den Vorteil, dass man bei allen nicht angegebenen Optionen auf die Standardeinstellungen zurückgreift. Damit ist man auch nicht gezwungen, sich bei einer Aktualisierung von Portage darum zu kümmern, ob die eigenen Einstellungen noch Gültigkeit besitzen oder ob sich das Konfigurationsformat geändert hat. Über diese Grundeinstellungen hinaus sind die folgenden Anpassungen für den Anfänger sinnvoll:

- Die USE-Flags auf die eigenen Bedürfnisse anpassen (siehe Kapitel 5.1 ab Seite 114).
- Die `SYNC`-Variable anpassen und einmal `mirrorselect -o -s5 >> /etc/make.conf` laufen lassen (siehe Kapitel 6.2 ab Seite 143).
- Die Variable `FEATURES` auf `"sandbox parallel-fetch strict distlocks"` setzen (siehe Kapitel 6.3.3 ab Seite 150).

Am Ende sollte sich eine Datei ergeben, die nicht viel komplizierter ist als die oben angegebene:

```
gentoo ~ # cat /etc/make.conf
CFLAGS="-march=i686 -O2 -pipe"
CXXFLAGS="${CFLAGS}"
CHOST="i686-pc-linux-gnu"
USE="apache2 mysql xml ldap -X"

SYNC="rsync://rsync.europe.gentoo.org/gentoo-portage"
GENTOO_MIRRORS="ftp://pandemonium.tiscali.de/pub/gentoo/ http://gentoo.m
neisen.org/ ftp://sunsite.informatik.rwth-aachen.de/pub/Linux/gentoo ftp
://ftp.free.fr/mirrors/ftp.gentoo.org/ http://213.186.33.38/gentoo-distf
iles/"

FEATURES="sandbox parallel-fetch strict distlocks"
```

Kapitel 7

Das Init-System

Mit dem Herzstück von Gentoo, dem Portage-System, haben wir uns jetzt ausgiebig vertraut gemacht und wir sollten problemlos in der Lage sein, neue Software exakt nach unseren Wünschen im System einzuspielen. Kein Linux-System besteht aber nur aus dem Paketmanagement-System, sondern bietet eine Vielzahl weiterer Konfigurationsmöglichkeiten. Die Kernel-Konfiguration und die Netzwerkeinstellungen haben wir schon in Kapitel 2 und 3 beleuchtet, um in der Lage zu sein, neue Software herunter zu laden.

In den folgenden zwei Kapiteln vertiefen wir die Konfiguration des Boot-Prozesses und der Lokalisation, um das Thema Systemeinstellungen dann in Kapitel 9 mit einer Sammlung kleinerer, aber wichtigen, Konfigurationsoptionen abzuschließen.

Um zu verstehen, wie der Systemstart unter Gentoo abläuft und wie wir ihn als Nutzer beeinflussen können, dienen die Abschnitte 7.1 bis 7.3. Die Funktionsweise der Init-Skripte vertiefen wir dann ab Seite 174, wobei dieser Teil für die Erstkonfiguration unseres Systems nicht notwendig ist.

7.1 Runlevel

Jedes Rechner-System durchläuft beim Booten eine spezifische Initialisierungssequenz, während der eine Reihe von Skripten dafür sorgt, dass die Maschine zuletzt die volle Funktionalität bietet.

Die Skripte für die Initialisierung eines Gentoo-Systems befinden sich Linux-Standard-Base-konform im Ordner `/etc/init.d`.

```
gentoo ~ # ls /etc/init.d/
apache2      depscan.sh    localmount    numlock       spind
autoconfig   functions.sh  modules       reboot.sh     sshd
bootmisc     gpm           mysql         rmnologin     syslog-ng
checkfs      halt.sh       mysqlmanager  rsyncd        urandom
checkroot    hdparm        net.eth0      runscript.sh  vixie-cron
clock        hostname      net.lo        shutdown.sh
consolefont  keymaps       netmount      slapd
crypto-loop  local         nscd          slurpd
```

Es liegt jedoch beim Nutzer, welche dieser Skripte (und damit welche Dienste) er gestartet sehen möchte. Anders als bei Distributionen, die das System-V-Init-System benutzen – und das sind derzeit (noch) die meisten –, dient bei Gentoo das Verzeichnis `/etc/runlevels` der Auswahl der zu startenden Dienste.

Wie der Name vermuten lässt, werden hier die verschiedenen Runlevels des Systems definiert. Anders als von anderen Distributionen gewohnt, identifiziert Gentoo diese über lesbare Namen. Die Datei `/etc/inittab` verknüpft den Runlevel-Namen mit der bekannten Nummerierung von 0 bis 6:

```
gentoo ~ # cat /etc/inittab
...
rc::bootwait:/sbin/rc boot

l0:0:wait:/sbin/rc shutdown
l1:S1:wait:/sbin/rc single
l2:2:wait:/sbin/rc nonetwork
l3:3:wait:/sbin/rc default
l4:4:wait:/sbin/rc default
l5:5:wait:/sbin/rc default
l6:6:wait:/sbin/rc reboot
...
```

`man inittab` erklärt die Details dieser Einträge; an dieser Stelle reicht ein Blick auf den zweiten und den letzten Eintrag in jeder Zeile (als Trenner dient der Doppelpunkt). Die Ziffer in der zweiten Spalte bezeichnet den numerischen Runlevel, der Befehl in der letzten Spalte die zu startende Aktion.

`/sbin/rc` dient unter Gentoo als Handler für das Init-System; das ihm übergebene Argument bezeichnet den Namen des zu startenden Runlevels innerhalb von `/etc/runlevels`.

So wird der Runlevel `boot` aufgrund des fehlenden Eintrags in der zweiten Spalte für jeden der numerischen Runlevel von 0 bis 6 ausgeführt. Beim Wechsel in den Runlevel 3, 4 oder 5 starten zusätzlich die Dienste, die für den Gentoo-Runlevel `default` vorgesehen sind.

Den numerischen Runlevel können wir unter Gentoo vergleichbar zu anderen Distributionen über die Kernel-Parameter beim Booten festlegen. Hierfür hängt man die entsprechende Zahl an den Boot-Eintrag in der Grub-Konfiguration an (siehe Seite 58).

Um den Runlevel 4 über das Boot-Menü auszuwählen, kann z. B. folgender Eintrag dienen:

```
title=Gentoo Linux (Runlevel 4)
root (hd0,0)
kernel /kernel root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root=/dev\
/hda3 udev 4
initrd /initramfs
```

Angenommen wir hätten unter `/etc/runlevels` einen neuen Runlevel mit dem Namen `graphical` erstellt, der uns nicht in die Kommandozeile befördert, sondern den X-Server startet und eine grafische Benutzeroberfläche bereitstellt. Dann könnten wir diesen jetzt mit dem numerischen Runlevel 4 verknüpfen, indem wir die entsprechende Zeile in der `/etc/inittab` folgendermaßen modifizieren:

```
l4:4:wait:/sbin/rc graphical
```

Die Assoziation über die Nummern und den Umweg über die Datei `/etc/inittab` sind allerdings ein wenig umständlich; man kann sie sich sparen, indem man statt der Zahl den Parameter `softlevel=`*`runlevel`* an die Bootparameter anhängt.

```
title=Gentoo Linux (grafische Benutzeroberfläche)
root (hd0,0)
kernel /kernel root=/dev/ram0 init=/linuxrc ramdisk=8192 real_root=/dev\
/hda3 udev softlevel=graphical
initrd /initramfs
```

Modifikationen an `inittab` sind in diesem Fall nicht notwendig, da das `/sbin/rc`-Skript den angegebenen Wert `default` ignoriert und den unter `softlevel` angegebenen Runlevel auswählt.

Beim Wechsel in den entsprechenden Runlevel führt `/sbin/rc` die Skripte innerhalb des entsprechenden Unterverzeichnisses von `/etc/runlevels`

aus. Diese Runlevel-Verzeichnisse enthalten allerdings keine Kopien der Bootskripte, stattdessen verweisen Links auf die entsprechenden Original-Skripte in /etc/init.d:

```
gentoo ~ # ls -la /etc/runlevels/boot/
insgesamt 8
drwxr-xr-x 2 root root 4096 20. Apr 2007  .
drwxr-xr-x 6 root root 4096 20. Apr 2007  ..
lrwxrwxrwx 1 root root   20 21. Jan 15:56 bootmisc -> /etc/init.d/bootmi
sc
lrwxrwxrwx 1 root root   19 21. Jan 15:56 checkfs -> /etc/init.d/checkfs
lrwxrwxrwx 1 root root   21 21. Jan 15:56 checkroot -> /etc/init.d/check
root
lrwxrwxrwx 1 root root   17 21. Jan 15:56 clock -> /etc/init.d/clock
lrwxrwxrwx 1 root root   23 21. Jan 15:56 consolefont -> /etc/init.d/con
solefont
lrwxrwxrwx 1 root root   20 21. Jan 15:56 hostname -> /etc/init.d/hostna
me
lrwxrwxrwx 1 root root   19 21. Jan 15:56 keymaps -> /etc/init.d/keymaps
lrwxrwxrwx 1 root root   22 21. Jan 15:56 localmount -> /etc/init.d/loca
lmount
lrwxrwxrwx 1 root root   19 21. Jan 15:56 modules -> /etc/init.d/modules
lrwxrwxrwx 1 root root   18 21. Jan 15:56 net.lo -> /etc/init.d/net.lo
lrwxrwxrwx 1 root root   21 21. Jan 15:56 rmnologin -> /etc/init.d/rmnol
ogin
lrwxrwxrwx 1 root root   19 21. Jan 15:56 urandom -> /etc/init.d/urandom
```

7.2 rc-update

Um den Nutzern die Arbeit zu ersparen, die entsprechenden Links manuell zu setzen, bietet Gentoo das Tool `rc-update`, das wir auf Seite 52 erstmalig verwendet haben. Es kennt drei Arbeitsmodi.

`add`
: fügt den angegebenen Runlevels ein Skript aus /etc/init.d hinzu:

```
rc-update add skriptname runlevel1 runlevel2 ...
```

`del`
: entfernt ein Skript aus allen bzw. nur den angegebenen Runlevels:

```
rc-update del skriptname
rc-update del skriptname runlevel1 runlevel2 ...
```

`show`
: zeigt an, welche Skripte in den angegebenen Runlevels aktiv sind:

```
rc-update show runlevel1 runlevel2 ...
```

Hier als Beispiel noch einmal die aktiven Skripte im Runlevel `boot`:

```
gentoo ~ # rc-update show boot
            bootmisc | boot
             checkfs | boot
           checkroot | boot
               clock | boot
         consolefont | boot
            hostname | boot
             keymaps | boot
          localmount | boot
             modules | boot
              net.lo | boot
           rmnologin | boot
             urandom | boot
```

Gibt man `show` die Option `--verbose` mit auf den Weg, zeigt `rc-update` nicht nur die Skripte an, die sich in den ausgewählten Runlevels befinden, sondern alle vorhandenen Skripte.

Übrigens gibt es die Möglichkeit, die gleichen Aktionen über das Programm `eselect` auszuführen. Genaueres zur Installation und dem Werkzeug selbst findet sich in Kapitel 11.4 ab Seite 263. Das Runlevel-Modul für `eselect` heißt `rc` und ein Init-Skript fügen wir mit `eselect rc add` zu den angegebenen Runleveln hinzu. `eselect rc delete` löscht ein Skript aus den angegebenen Runlevels, ist also vergleichbar zu `rc-update del` und schließlich zeigt `eselect rc show` vergleichbar zu `rc-update show` die Einträge in den ausgewählten Runlevels.

```
gentoo ~ # eselect rc add apache2 default
Adding apache2 to following runlevels
  default                  [done]
gentoo ~ # eselect rc show default
Status of init scripts in runlevel default
Status of init scripts in runlevel default
  apache2                  [stopped]
  local                    [started]
  net.eth0                 [started]
  netmount                 [started]
  syslog-ng                [started]
  vixie-cron               [started]
gentoo ~ # eselect rc delete apache2 default
Deleting apache2 from following runlevels
  default                  [done]
```

Die Ausgabe sieht etwas anders aus, der Effekt ist aber derselbe. Langjährige Gentoo-Benutzer bleiben vermutlich eher `rc-update` verhaftet, aber da sich `eselect` immer mehr durchsetzt, ist auch dieses Werkzeug für das Management der Runlevels zu empfehlen.

7.2.1 Die Konfiguration des Init-Systems

Die meisten Init-Skripte akzeptieren einige Konfigurationsvariablen. Diese befinden sich grundsätzlich im Verzeichnis `/etc/conf.d` und tragen den gleichen Namen wie das entsprechende Init-Skript. Die Konfiguration für `/etc/init.d/apache2` findet sich also in `/etc/conf.d/apache2`. Die Konfigurationsdateien in `/etc/conf.d` sind alle recht gut kommentiert, und in vielen Fällen ist die vorgegebene Standard-Konfiguration ausreichend.

Die entsprechenden Optionen liest das Init-System beim Start eines Services ein und stellt sie innerhalb des Init-Skripts zur Verfügung. Abgesehen von den Konfigurationswerten in `/etc/conf.d/`*SERVICE* gilt das auch für die Parameter der Dateien `/etc/conf.d/rc` und `/etc/rc.conf`. Eine genauere Übersicht findet sich bei den Servicevariablen im Abschnitt 9.2 ab Seite 195.

7.3 Verwendung von Init-Skripten unter Gentoo

Die Init-Skripte lassen sich auch separat ansprechen, um einzelne Dienste im laufenden System zu starten oder zu beenden. So können wir z. B. den Apache-Server über den Befehl `/etc/init.d/apache2 start` hochfahren:

```
gentoo ~ # /etc/init.d/apache2 start
 * Starting apache2 ...                                [ ok ]
```

Folgende Kommandos sind für ein Init-Skript immer möglich:

```
start           zap             needsme
stop            status          useme
restart         ineed           broken
pause           iuse
```

Beschäftigen wir uns zunächst einmal mit den gebräuchlichsten Kommandos, die den Zustand des Service betreffen: `start`, `stop`, `restart`, `pause`, `zap`, `status`.

Wie wir schon gesehen haben, lässt sich ein Service über den `start`-Befehl hochfahren, ein gestoppter Service auch über das `restart`-Kommando wieder starten. Befand sich der Service schon im gestarteten Zustand, so wird `restart` ihn kurz unterbrechen und wieder in den laufenden Status befördern. Wir testen das im Folgenden mit Hilfe der Prozessliste anhand des laufenden Apache-Servers:

```
gentoo ~ # /etc/init.d/apache2 restart
 * Stopping apache2 ...                                [ ok ]
 * Starting apache2 ...                                [ ok ]
gentoo ~ # ps ax | grep apache
12984 ?        Ss     0:00 /usr/sbin/apache2 ...
12991 ?        S      0:00 /usr/sbin/apache2 ...
12992 ?        S      0:00 /usr/sbin/apache2 ...
12993 ?        S      0:00 /usr/sbin/apache2 ...
12994 ?        S      0:00 /usr/sbin/apache2 ...
12995 ?        S      0:00 /usr/sbin/apache2 ...
13028 pts/0    S+     0:00 grep --colour=auto apache
```

Das Init-System führt keine direkte Überprüfung des Service-Zustands in der Form durch, wie wir es hier über `ps` getan haben. Im Normalfall überpüft es beim Start-Vorgang über das Init-Skript die Rückgabewerte der gestarteten Server-Prozesse. Sollten hier Fehler auftreten, wird der Benutzer gewarnt und der Service nicht als gestartet markiert. Sollte ein Service jedoch einwandfrei starten und dann im laufenden Betrieb zusammenbrechen, wird das dem Init-System nicht auffallen.

Wir wollen das einmal testen, indem wir den Apache-Prozess gezielt unterbrechen. Zunächst überprüfen wir den Zustand des Service über das `status`-Kommando:

```
gentoo ~ # /etc/init.d/apache2 status
 * status:  started
```

Nun hintergehen wir das Init-System, töten den Apache-Prozess und überprüfen nochmals den Status:

```
gentoo ~ # killall apache2
gentoo ~ # /etc/init.d/apache2 status
 * status:  started
```

Wir sehen, dass unser Init-Skript immer noch der Überzeugung ist, dass der Service läuft. Ein Check der Prozessliste belehrt uns aber eines Besseren:

```
gentoo ~ # ps ax | grep apache
13077 pts/0    S+     0:00 grep --colour=auto apache
```

An diesem Punkt haben wir ein kleines Problem, denn wenn wir nun versuchen, den Service erneut zu starten, hält uns das Init-System davon ab:

```
gentoo ~ # /etc/init.d/apache2 start
 * WARNING:  apache2 has already been started.
```

Wir erhalten die Warnung, der Service sei schon gestartet, und da Init dieser Überzeugung ist, versucht es erst gar keinen Neustart. In dieser Situation benötigen wir das `zap`-Kommando, um Init von dem wirklichen Status des Systems zu überzeugen:

```
gentoo ~ # /etc/init.d/apache2 zap
 * Manually resetting apache2 to stopped state.
```

Wir haben den Status nun manuell zurück gesetzt und können auch den `start`-Befehl wieder verwenden:

```
gentoo ~ # /etc/init.d/apache2 start
 * Starting apache2 ...                                [ ok ]
gentoo ~ # ps ax | grep apache
13170 ?        Ss     0:00 /usr/sbin/apache2 ...
13173 ?        S      0:00 /usr/sbin/apache2 ...
13174 ?        S      0:00 /usr/sbin/apache2 ...
13175 ?        S      0:00 /usr/sbin/apache2 ...
13176 ?        S      0:00 /usr/sbin/apache2 ...
13177 ?        S      0:00 /usr/sbin/apache2 ...
13179 pts/0    S+     0:00 grep --colour=auto apache
```

Bleibt an dieser Stelle noch das `pause`-Kommando, mit dem wir auch gleich zum nächsten Punkt überleiten können: Abhängigkeiten zwischen verschiedenen Services. Fahren wir dafür einmal bei gestartetem Apache-Server das Netzwerk herunter:

```
gentoo ~ # /etc/init.d/net.eth0 stop
 * Stopping apache2 ...                                [ ok ]
 * Stopping eth0
 *   Bringing down eth0
 *     Shutting down eth0 ...                           [ ok ]
```

In dieser Situation beendet das Init-System nicht nur das Netzwerk, sondern auch den Apache-Server. Es muss also einen Mechanismus geben, der dem Init-System unter Gentoo mitteilt, dass der Apache eine funktionierende Netzwerkschnittstelle benötigt, um korrekt zu funktionieren. Auf diese Art der Abhängigkeiten gehen wir im nächsten Abschnitt genauer ein.

An dieser Stelle wollen wir uns aber erst einmal ansehen, wie wir diese Abhängigkeiten umgehen können. Es gibt Situationen, in denen wir einen Service nur kurz unterbrechen wollen, um ihn anschließend wieder zu starten, und wir trotzdem nicht gleich die ganze Kette aller abhängigen Systeme ebenfalls beenden wollen. Hier hilft der Befehl `pause`. Zur Demonstration starten wir den Apache bei abgeschaltetem Netzwerk erst einmal wieder:

```
gentoo ~ # /etc/init.d/apache2 start
 * Starting apache2 ...                                [ ok ]
 * Starting eth0
 *   Bringing up eth0
 *     dhcp
 *       Running dhcpcd ...
Error, timed out waiting for a valid DHCP response    [ !! ]
```

```
 *     Trying fallback configuration
 *     192.168.178.66                                  [ ok ]
 * Starting apache2 ...                                [ ok ]
```

Wir sehen das umgekehrte Verhalten von vorhin: Da der Apache eine korrekt konfigurierte Netzwerkschnittstelle benötigt, startet das Init-System hier zuerst `net.eth0`, um dann im zweiten Schritt den Apache-Server zu initialisieren.

Verwenden wir jetzt `pause` anstatt `stop`, um das Netzwerk herunter zu fahren, achtet das Init-System nicht weiter auf die Abhängigkeiten und der Apache-Server läuft weiter.

```
gentoo ~ # /etc/init.d/net.eth0 pause
 * Stopping eth0
 *   Bringing down eth0
 *     Shutting down eth0 ...                          [ ok ]
```

Starten können wir den Service dann wieder wie üblich mit `start`:

```
gentoo ~ # /etc/init.d/net.eth0 start
 * Starting eth0
 *   Bringing up eth0
 *     dhcp
 *       Running dhcpcd ...
Error, timed out waiting for a valid DHCP response    [ !! ]
 *     Trying fallback configuration
 *     192.168.178.66                                  [ ok ]
```

Bleiben noch zwei Optionen zu erwähnen, die man jedem Init-Skript mit auf den Weg geben kann: `--nocolor` und `--quiet`. `--nocolor` dient dazu, Farbwechsel aus der Kommandozeilenausgabe zu entfernen, so dass die Meldungen auch auf Terminals ohne Farbunterstützung lesbar sind. `--quiet` reduziert die Ausgabe des Befehls auf Warnungen und eventuelle Fehler. Im Normalfall sollte also keine Ausgabe zu sehen sein:

```
gentoo ~ # /etc/init.d/apache2 --quiet restart
gentoo ~ #
```

Übrigens kann, wer mag, auch für das Service-Management das Programm `eselect` verwenden:

```
gentoo ~ # eselect rc restart apache2
Restarting init script
 * Stopping apache2 ...        [ ok ]
 * Starting apache2 ...        [ ok ]
```

Die Befehle `start`, `stop`, `restart` und `pause` haben den gleichen Effekt, als würde man das Init-Skript wie oben beschrieben direkt aufrufen. Nur der `status`-Befehl wurde umbenannt und heißt im `rc`-Modul `show`. Der Befehl `zap` fehlt für `eselect rc`.

Einen kleinen Vorteil hat die Verwendung von `eselect`, denn es lassen sich gleich mehrere Service-Namen angeben. `eselect rc restart apache2 mysql` startet also gleichzeitig den Apache-Server und die MySQL-Datenbank.

Wir vertiefen uns nun in die Funktionsweise der Init-Skripte. Wer sich statt dessen zunächst auf die Grundkonfiguration des Systems konzentrieren möchte, dem sei geraten, an diesem Punkt auf das nächste Kapitel ab Seite 179 zu springen.

7.3.1 Abhängigkeiten

Wie funktionieren die oben angesprochenen Abhängigkeiten? Dafür müssen wir uns ein Init-Skript einmal etwas genauer ansehen. Hierfür bietet sich `/etc/init.d/apache2` an. Hier die ersten Zeilen des Skripts:

```
#!/sbin/runscript
# Copyright 1999-2005 Gentoo Foundation
# Distributed under the terms of the GNU General Public License v2

opts="${opts} reload configtest"

# this next comment is important, don't remove it - it has to be
# somewhere in the init script to kill off a warning that doesn't
# apply to us
# svc_start svc_stop

depend() {
        need net
        use mysql dns logger netmount postgresql
        after sshd
}

...
```

Wichtig ist an dieser Stelle die Definition von `depend`. Hier sehen wir auch schon, was in den vorigen Beispielen dazu geführt hat, dass Init die Netzwerkschnittstelle für den Apache-Server als notwendig erachtet hat: der Befehl `need net`.

Statt direkt im Skript nach zuschauen, können wir das Init-Skript auch mit dem Befehl `ineed` nach zwingenden Abhängigkeiten befragen:

```
gentoo ~ # /etc/init.d/apache2 ineed
net
```

Umgekehrt ist es auch möglich, das Netzwerkskript danach zu befragen, welche Services von diesem Element abhängen:

```
gentoo ~ # /etc/init.d/net.eth0 needsme
apache2 netmount slapd slurpd sshd net
```

Die Sammlung an Services, die das Netzwerk benötigen, ist schon etwas größer. Der Apache-Server findet sich direkt an erster Stelle wieder.

Wir sehen innerhalb des oben angegebenen `depend`-Statements aber auch noch den Begriff `use`, eine Zeile unter der `need`-Deklaration. `use` (benutzen) klingt weniger zwingend als `need` (benötigen), und das macht auch den Unterschied der beiden Deklarationen aus. Für den Apache-Server führt das entsprechende Init-Skript mehrere Dienste in der `use`-Liste:

```
use mysql dns logger netmount postgresql
```

Hier befindet sich z. B. der MySQL-Server. Nun läuft ein Apache-Server aber völlig problemlos ohne eine MySQL-Datenbank. Es ist aber natürlich so, dass viele Webanwendungen, vor allem solche, die auf PHP basieren, recht häufig eine MySQL-Datenbank verwenden.

Will man solche Anwendungen auf einem Webserver installieren, so liegt es nahe, neben dem Apache-Server auch den MySQL-Server zu starten. Um den Benutzer nicht zu bevormunden, benutzt das Init-System hier folgendes Verfahren: Wenn der unter `use` angegebene Dienst im derzeit aktiven Runlevel als aktiv markiert ist – also über `rc-update add` in den entsprechenden Ordner unter `/etc/runlevels` eingetragen wurde –, führt der Start eines Dienstes auch zum Start des entsprechenden Subsystems.

Probieren wir das einmal mit MySQL aus. Dafür gehen wir sicher, dass sowohl der MySQL-Server als auch der Apache gestoppt sind und der `mysql`-Eintrag im derzeitigen Runlevel (hier `default`) fehlt.

```
gentoo ~ # /etc/init.d/mysql stop
 * WARNING:  mysql has not yet been started.
gentoo ~ # /etc/init.d/apache2 stop
 * Stopping apache2 ...                                [ ok ]
gentoo ~ # eselect rc delete mysql default
Deleting mysql from following runlevels
  default                  [skipped]
```

Wir überprüfen noch einmal mittels `iuse`, ob der Apache denn MySQL wirklich verwenden würde, und starten dann den Apache:

```
gentoo ~ # /etc/init.d/apache2 iuse
localmount net netmount mysql syslog-ng
gentoo ~ # /etc/init.d/apache2 start
 * Starting apache2 ...                                [ ok ]
```

Wie oben beschrieben, fehlt MySQL im aktuellen Runlevel, also wird der Service hier nicht gestartet. Ergänzen wir den Runlevel und starten den Apache erneut:

```
gentoo ~ # eselect rc add mysql default
Adding mysql to following runlevels
  default                  [done]
gentoo ~ # /etc/init.d/apache2 restart
 * Stopping apache2 ...                                [ ok ]
 * Starting mysql ...
 * Starting mysql (/etc/mysql/my.cnf)                  [ ok ]
 * Starting apache2 ...                                [ ok ]
```

Da wir dem System mitgeteilt haben, dass wir den MySQL-Server beim Boot-Vorgang im `default`-Runlevel gerne gestartet hätten, geht das Init-System nun davon aus, dass es den Server auch dann zur Verfügung stellen soll, wenn ein Service diesen als `use` markiert hat. Wie es für `needsme` den Counterpart `ineed` gibt, so gehören `usesme` und `iuse` zusammen.

```
gentoo ~ # /etc/init.d/mysql usesme
apache2
```

Der einzige Befehl, den wir an dieser Stelle noch nicht erwähnt haben, ist `broken`. Dieser lässt sich zusammen mit einem Init-Skript verwenden, um zu überprüfen, ob irgendwelche Abhängigkeiten, die über `need` spezifiziert wurden, nicht erfüllt sind.

Im Normalfall liefert dieser Test jedoch kein Ergebnis, da schon der Ebuild überprüft, ob alle notwendigen Abhängigkeiten erfüllt sind (siehe Seite 98). Wenn also ein Paket im Init-Skript die MySQL-Datenbank mit `need` spezifiziert, muss der Ebuild ebenfalls die Installation der MySQL-Datenbank zwingend voraussetzen. Die Situation, die sich über `broken` überprüfen lässt, können wir also eigentlich nur erzeugen, indem wir ein Paket unbedacht über `emerge --unmerge` entfernen (siehe Kapitel 4.3.5). Entsprechend selten findet der Befehl Verwendung und liefert, wie hoffentlich auch in unserem System, keine Ausgabe:

```
gentoo ~ # /etc/init.d/apache2 broken

gentoo ~ #
```

Wer genau hin geschaut hat, findet in der oben angegebenen `depend`-Sektion noch den Befehl `after`. Zu dieser Deklaration gibt es auch den Gegenspieler `before`. Beide werden eingesetzt, um die Start-Reihenfolge der verschiedenen Services festzulegen. Dies geschieht bei den meisten Distributionen über spezifische Start-Nummern, die den Skripten vorangestellt werden – meist zwischen `00` und `99`.

Unter Gentoo erreicht man das gleiche Ziel über die direkte Definition mit `before` und `after`. Der Vorteil ist klar: Der Entwickler eines Init-Skripts muss nur die Services definieren, die wirklich vor oder nach dem Ziel des Init-Skripts gestartet werden müssen, und kann diese direkt formulieren, ohne sich um die Sortierung einer globalen Liste kümmern zu müssen.

Um die Auflösung der zeitlichen Abhängigkeiten kümmert sich dann das spezielle Init-System von Gentoo. Für den Apache-Service finden wir in dem Init-Skript nur eine `after`-Deklaration:

```
after sshd
```

Das Init-System fährt also den Apache-Server immer erst nach dem SSH-Server hoch.

Eine letzte spezielle Deklaration, die im Init-Skript des Apache-Servers fehlt, aber z. B. in dem Netzwerkskript `/etc/init.d/net.eth0` zu finden wäre, ist die `provide`-Direktive. Diese ist für die besondere Situation gedacht, dass verschiedene Services ein und dieselbe Funktionalität bieten. So haben wir ja weiter oben schon gesehen, dass die notwendige Netzwerkschnittstelle nicht über `use net.eth0`, sondern mit Hilfe von `use net` definiert wurde. Das ist auch sinnvoll, da ja eine WLAN-Schnittstelle, die vielleicht über `/etc/init.d/net.wlan0` gestartet würde, durchaus die gleiche Netzwerkfunktionalität bieten würde wie `net.eth0`. In diesem Fall muss also jedes Netzwerkskript über `provide net` deklarieren, dass es ganz allgemein den Zugang zu einem Netzwerk ermöglicht. Die gleiche Situation gibt es z. B. auch beim Syslog- (`provide logger`) oder dem Cron-System (`provide cron`).

Kapitel

8 Lokalisierung

Wenn man sich eingehender mit der *Lokalisierung,* also der Anpassung des Rechners an lokale Gegebenheiten (Uhrzeit, Sprache etc.) beschäftigt, überrascht der Stand der Entwicklung bisweilen sehr. Vor dem Hintergrund des weltumspannenden Internets sollte man erwarten, dass sich Rechner, Betriebssysteme und Programme stärker an einheitliche Standards halten.

In vielen Bereichen hapert es jedoch mit der Kompatibilität, und so erblicken Benutzer weltweit wohl noch einige Zeit gelegentlich seltsame Hieroglyphen auf ihren Bildschirmen und fragen sich, was da bei der Konvertierung der Zeichensätze schief gelaufen ist.

Allerdings fördert eben vor allem das Internet und der Boom der Kommunikationsbranche, dass neue Standards geschaffen und auch mehr und mehr eingehalten werden.

Wir wollen an dieser Stelle versuchen, unser Gentoo-System möglichst standardkonform einzurichten, und an einigen Stellen auch auf mögliche Probleme hinweisen.

Wer sich nach der Erstinstallation zunächst einmal nicht um das korrekte Tastaturlayout kümmern oder Fehlermeldungen dringend in Deutsch erhalten möchte, der kann natürlich auch mit Kapitel 9 fortfahren und hierher zurückkehren, wenn er mit dem System doch in der eigenen Muttersprache reden möchte.

8.1 Uhrzeit

Die Einstellungen zur Uhrzeit erfolgen unter `/etc/conf.d/clock`.

```
gentoo ~ # cat /etc/conf.d/clock
# /etc/conf.d/clock

# Set CLOCK to "UTC" if your system clock is set to UTC (also known as
# Greenwich Mean Time).  If your clock is set to the local time, then
# set CLOCK to "local".  Note that if you dual boot with Windows, then
# you should set it to "local".

CLOCK="UTC"

# Select the proper timezone.  For valid values, peek inside of the
# /usr/share/zoneinfo/ directory.  For example, some common values are
# "America/New_York" or "EST5EDT" or "Europe/Berlin".

#TIMEZONE="Factory"

# If you wish to pass any other arguments to hwclock during bootup,
# you may do so here.

CLOCK_OPTS=""

# If you want to set the Hardware Clock to the current System Time
# during shutdown, then say "yes" here.

CLOCK_SYSTOHC="no"
```

8.1.1 Hardware-Uhr

Jeder Rechner besitzt eine batteriebetriebene Hardware-Uhr, die auch über das Ausschalten des Rechners hinaus ihren Dienst tut. Diese Uhr lässt sich mit dem Programm `hwclock` einstellen.

Linux setzt diese Uhr normalerweise auf die aktuelle Zeit, entsprechend der koordinierten Weltzeit (UTC). UTC ist eine eindeutige und vor allem standardisierte Referenzzeit. In Deutschland sowie den meisten westeuropäischen Ländern entspricht die Uhrzeit UTC+1, also eine Stunde später als durch UTC angegeben.

Diesen, dem momentanen Standort entsprechenden zeitlichen Versatz legt die Datei /etc/localtime fest. Diese Datei haben wir bei der Installation (siehe Kapitel 1.10 ab Seite 46) angelegt und dem System damit mitgeteilt, wie es die Zeit errechnen soll. Wechselt man z. B. mit seinem Laptop den Standort, passt man /etc/localtime entsprechend an. Die Uhrzeit der Hardware-Uhr wird also durch den Ortswechsel nicht berührt.

Das handhabt allerdings nicht jedes Betriebssystem so. Speziell Windows setzt die Hardware-Uhr direkt auf die lokale Zeit. Will man seinen Rechner unter beiden Betriebssystemen betreiben, muss man die CLOCK-Standardeinstellung UTC auf local umstellen. Es gibt keine Möglichkeit, automatisch zu erkennen, auf welche Zeitzone die Hardware-Uhr eingestellt ist. Hier ist man also gezwungen, selbst die richtige Einstellung zu wählen.

In der Standardeinstellung liest Gentoo die Einstellung der Hardware-Uhr nur, schreibt sie jedoch nicht. Damit wird das Risiko minimiert, andere Betriebssysteme auf dem Rechner zu stören. Wer Gentoo alleine auf dem Rechner betreibt oder sich sicher ist, dass die Konfiguration für mehrere Betriebssysteme stimmt, sollte CLOCK_SYSTOHC auf yes setzen, um die Systemzeit beim Herunterfahren des Rechners in die Hardware-Uhr zu schreiben und dabei die Ungenauigkeit der Uhr zu bestimmen. Damit verbessert sich dann die Genauigkeit der Zeitmessung.

Lesen und Schreiben der Hardware-Uhr ist in jedem Fall durch Skripte gekapselt und im Normalfall gibt es keinen Grund, sich direkt mit hwclock auseinander zu setzen. Deshalb demonstrieren wir an dieser Stelle nur den Befehl, über den sich die Einstellung der Hardware-Uhr anzeigen lässt:

```
gentoo ~ # /sbin/hwclock --show
Mo 02 Apr 2007 10:18:51 CEST  -0.759538 Sekunden
```

Die am Ende angezeigten Sekunden informieren über die Zeit, die zwischen dem Programmaufruf und der Anzeige der Zeit vergangen sind.

Ob hwclock die Hardware-Uhr als UTC oder die lokale Zeit wahrnimmt, findet sich dann in der Datei /etc/adjtime wieder:

```
gentoo ~ # cat /etc/adjtime
-1.839798 1175495815 0.000000
1175447942
UTC
```

Wer sich doch intensiver mit den Funktionen von hwclock auseinandersetzen und dem Werkzeug besondere Optionen beim Boot-Vorgang mit auf den Weg geben möchte, kann dazu die Variable CLOCK_OPTS verwenden.

8.1.2 Systemzeit

Die Systemzeit setzen wir, wie bereits erwähnt, über `/etc/localtime`. Um die Zeit des Rechners anzupassen, reicht es, wie schon bei der Installation beschrieben, die entsprechende Datei aus `/usr/share/zoneinfo/` nach `/etc/localtime` zu kopieren.

```
gentoo ~ # cp /usr/share/zoneinfo/Europe/Berlin /etc/localtime
```

Gleichzeitig sollte man aber auch die Einstellung `TIMEZONE` in `/etc/conf.d/clock` anpassen. Für Deutschland ist die korrekte Einstellung `Europe/Berlin` und entspricht damit dem oben angegebenen Dateinamen.

Nur das Paket `sys-libs/timezone-data` verwendet diese Einstellung. Es installiert die Zonen-Informationen unterhalb von `/usr/share/zoneinfo/` (aus dem wir ja auch unsere derzeitige `/etc/localtime`-Datei kopiert haben) und verwendet die Information aus `TIMEZONE`, um bei einer Aktualisierung des Pakets die Datei `/etc/localtime` korrekt zu aktualisieren.

`/etc/localtime` beeinflusst die Systemzeit und damit den Standardwert, der für jeden Benutzer gilt.

8.1.3 Benutzerzeit

Für einen einzelnen Benutzer lässt sich die angezeigte Zeit über die Variable `TZ` beeinflussen.

```
gentoo ~ # export TZ="Indian/Cocos"
gentoo ~ # date
Mo Apr  2 21:37:42 CCT 2007
gentoo ~ # export TZ="Europe/Berlin"
gentoo ~ # date
Mo Apr  2 17:07:47 CEST 2007
```

Das obige Beispiel informiert uns über die aktuelle Uhrzeit in Indien und wechselt dann wieder auf unsere Standardeinstellungen zurück.

Nutzer, die dauerhaft eine von der Systemzeit abweichende Zeitzone festlegen wollen, können den entsprechenden Wert für `TZ` in ihrem Benutzerverzeichnis in `~/.bash_profile` festlegen.

8.2 Tastatur

Die Einstellungen zum Tastaturlayout sind – unter Linux – bei der textbasierten Konsole und der grafischen Benutzeroberfläche unterschiedlich. Wir beschreiben hier nur die Einstellungen für die Konsole.

8.2.1 Tastatur für die Konsole

Das Tastaturlayout für die Konsole lässt sich in `/etc/conf.d/keymaps` festlegen.

```
gentoo ~ # cat /etc/conf.d/keymaps
# /etc/conf.d/keymaps

# Use KEYMAP to specify the default console keymap.  There is a complete
# tree of keymaps in /usr/share/keymaps to choose from.

KEYMAP="de-latin1-nodeadkeys"

# Should we first load the 'windowkeys' console keymap?  Most x86 users
# will say "yes" here.  Note that non-x86 users should leave it as "no".

SET_WINDOWKEYS="no"

# The maps to load for extended keyboards.  Most users will leave this
# as is.

EXTENDED_KEYMAPS=""
#EXTENDED_KEYMAPS="backspace keypad euro"

# Tell dumpkeys(1) to interpret character action codes to be
# from the specified character set.
# This only matters if you set UNICODE="yes" in /etc/rc.conf.
# For a list of valid sets, run `dumpkeys --help`

DUMPKEYS_CHARSET=""
```

Wer eine normale deutsche Tastatur verwendet, weist `KEYMAP` den Wert `de-latin1` zu oder, wenn man keine „toten“ Tasten wie ~ haben möchte, `de-latin1-nodeadkeys`. Tote Tasten liefern beim Anschlag kein Zeichen, sondern resultieren erst in der Kombination mit dem nächsten Anschlag in einem darstellbaren Zeichen. Ein Beispiel sind die französischen Vokale mit Akzent (é, è etc.).

Bei den `EXTENDED_KEYMAPS` sind im Normalfall keine weiteren Angaben notwendig. Die Keymap `euro` mag verführerisch klingen, aber `de-latin1` enthält bereits die Keymap `euro2`, die es einem erlaubt, mit der Kombination `[AltGr]+[e]` das Euro-Symbol auf den Schirm zu zaubern. Die Keymap `euro` verlegt diese Kombination auf `[Alt]+[e]` für Tastaturen, denen die Taste `[AltGr]` fehlt.

Sollte man versehentlich eine falsche Keymap geladen haben oder eine neue ausprobieren wollen, kann man das Programm `loadkeys` verwenden, um das Mapping im laufenden Betrieb zu verändern.

```
gentoo ~ # loadkeys de-latin1-nodeadkeys
```

Allerdings sollte man mit diesem Tool vorsichtig umgehen, denn mit einem völlig falschen Layout kann es schwierig werden, den Befehl zur Wiederherstellung des ursprünglichen Zustands zu tippen.

Wer sich genauer mit den verfügbaren Keymaps auseinander setzen möchte, dem sei das Studium des Verzeichnisses `/usr/share/keymaps` ans Herz gelegt. Außerdem sollte man sich die Hilfeseiten zu `loadkeys` und `keymaps` durchlesen. Allerdings ist die Definition dieser Keymaps unter Linux eine Wissenschaft für sich.

Für ein deutsches Layout (`de-latin1` oder `de-latin1-nodeadkeys`) ist keine Angabe unter `DUMPKEYS_CHARSET` notwendig. Verwendet man eine andere Keymap, so sollte man sich informieren, auf welchem Zeichensatz sie basiert und ob sie vom Standardwert `iso-8859-1` abweicht. Für die deutschen Layouts ist dies nicht der Fall.

8.2.2 Zeichensatz für die Konsole

Das Tastaturlayout sorgt dafür, dass unser System die Anschläge auch wirklich mit den Zeichen in Verbindung bringe, die sich als Beschriftung auf der Tastatur befinden. Für die Darstellung der Zeichen auf dem Bildschirm ist dann allerdings noch einmal ein Zeichensatz notwendig, der jedem Zeichen ein für den Benutzer lesbares Symbol zuordnet. Den geeigneten Zeichensatz für die Konsole legen wir in `/etc/conf.d/consolefont` fest:

```
gentoo ~ # cat /etc/conf.d/consolefont
# /etc/conf.d/consolefont

# CONSOLEFONT specifies the default font that you'd like Linux to use on
# the console.  You can find a good selection of fonts in /usr/share
# /consolefonts; you shouldn't specify the trailing ".psf.gz", just the
# font name below. To use the default console font, comment out the
# CONSOLEFONT setting below. This setting is used by the /etc/init.d
# /consolefont script (NOTE: if you do not want to use it, run
# "rc-update del consolefont" as root).

CONSOLEFONT="lat9-16"

# CONSOLETRANSLATION is the charset map file to use.  Leave commented
# to use the default one.  Have a look in /usr/share/consoletrans for a
# selection of map files you can use.

#CONSOLETRANSLATION="8859-1_to_uni"
```

Der Standard-Zeichensatz `default8x16` ist für den deutschen Sprachraum nur begrenzt geeignet. Die Zeichensätze für die Konsole sind in ihrer Auswahl beschränkt und können maximal 512 Zeichen abbilden. Für den deutschen Sprachraum ist der Zeichensatz `lat9-16` recht gut geeignet. Er unterstützt auch unter der Konsole das Euro-Symbol.

Hierbei wird allerdings ein wenig getrickst. Eigentlich legt das oben angesprochene Mapping `euro2` innerhalb des Tastaturlayouts `de-latin1` bzw. `de-latin1-nodeadkeys` fest, dass `[AltGr]+[e]` in dem Symbol `currency` resultieren soll. Für den westeuropäischen Raum ist hier also das Euro-Symbol durchaus angebracht, und `lat9-16` zeigt hier auch das entsprechende Symbol. Allerdings sieht das korrekte Symbol für `currency` gemäß Unicode so aus: (U+00A4). Unicode repräsentiert das Euro-Symbol dagegen durch den Wert U+20AC.

Leider existiert derzeit kein korrektes Tastaturlayout für die Euro-Unicode-Map in `/usr/share/keymaps`. Für Unicode-Fanatiker lässt sich die Situation korrigieren, indem man folgende zusätzliche Keymap mit `nano` als `/usr/share/keymaps/i386/include/euro-unicode.map` anlegt:

```
gentoo ~ # cat /usr/share/keymaps/i386/include/euro-unicode.map
altgr keycode 18 = U+20AC
```

Diese zusätzliche Keymap können wir dann in `/etc/conf.d/keymaps` unter `EXTENDED_KEYMAPS` eintragen:

```
EXTENDED_KEYMAPS="euro-unicode"
```

Damit lassen sich dann auch korrekte Unicode-Zeichensätze verwenden, so z. B. den Terminus-Schriftsatz, der einen guten Schnitt an darstellbaren Zeichen für westeuropäische Länder liefert. Er lässt sich über das Paket `media-fonts/terminus-font` installieren:

```
gentoo ~ # emerge -av media-fonts/terminus-font

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] media-fonts/terminus-font-4.20  USE="-X" 198 kB

Total: 1 package (1 new), Size of downloads: 198 kB

Would you like to merge these packages? [Yes/No]
```

Einer der enthaltenen Schriftsätze, `ter-v16b`, bietet einen normal großen Zeichensatz. `ter-v12n` dagegen einen besonders kleinen Schrifttyp.

Hinweis für Systeme mit Netzwerkanbindung

Um diesen speziellen Schriftsatz zu installieren, brauchen sie die Netzwerkverbindung *und* müssen ihr System bereits aktualisiert haben.

CONSOLETRANSLATION, die letzte Variable in /etc/conf.d/consolefont, muss man nicht festlegen, da wir mit Unicode arbeiten. Die entsprechende Zeile kann auskommentiert bleiben.

8.3 Lokalisierung der zentralen Systembibliothek glibc

Die zentrale Bibliothek eines Linux-Systems, die glibc, liefert auch verschiedene Werkzeuge für die Anpassung an die lokale Umgebung. Das betrifft nicht nur den Zeichensatz des Systems, sondern auch Spracheinstellungen, Messgrößen, sowie Einstellungen zu Währung und Papierformaten.

8.3.1 Lokalisierung systemweit festlegen

Die Einstellungen der gewünschten Umgebung fällt vor diesem Hintergrund dennoch denkbar einfach aus: Man trägt diese Umgebung in /etc/env.d/02locale ein und aktualisiert das System über env-update (siehe Seite 193):

```
gentoo ~ # cat /etc/env.d/02locale
LANG="de_DE.utf8"
LC_ALL="de_DE.utf8"
gentoo ~ # env-update
```

Diese Angabe wählt die deutsche Umgebung inklusive Unicode-Support. Auch wenn sicherlich noch nicht alle Programme eines Linux-Systems vollständig Unicode-kompatibel sind, ist dies mittlerweile die empfohlene Einstellung.

Die für LANG verfügbaren Werte lassen sich über locale -a anzeigen:

```
gentoo ~ # locale -a
C
de_DE
de_DE@euro
de_DE.utf8
en_US
en_US.utf8
POSIX
```

Die speziellen Varianten C und POSIX zeigt locale immer an, da sie die fest in die glibc-Bibliothek einkompilierte Grundeinstellung darstellen. Sie bezeichnen beide den durch den internationalen POSIX-Standard[1] festgelegten Grundsatz an Lokalisierungsinformationen.

[1] http://standards.ieee.org/regauth/posix/

Die anderen Werte entsprechen den verfügbaren Lokalisierungsdefinitionen innerhalb des `locale path`. Dieser ist standardmäßig auf `/usr/lib/locale` gesetzt, lässt sich aber auch mit dem Tool `localedef` ermitteln:

```
gentoo ~ # localedef --help | grep "locale path"
       locale path    : /usr/lib/locale:/usr/share/i18n
```

8.3.2 Lokalisierungen erstellen

Da jede Lokalisierung Speicherplatz verbraucht, erstellt Portage bei der Installation der `glibc` nicht alle möglichen Varianten automatisch. Der Benutzer kann die Liste der gewünschten Varianten selbst in der Datei `/etc/locale.gen` angeben (siehe Kapitel 1.16):

```
gentoo ~ # cat /etc/locale.gen
de_DE ISO-8859-1
de_DE@euro ISO-8859-15
de_DE.UTF-8 UTF-8
en_US ISO-8859-1
en_US.UTF-8 UTF-8
```

Die Datei verknüpft je eine Umgebungsdefinition aus `/usr/share/i18n/locales` mit einem Zeichensatz aus `/usr/share/i18n/charmaps`. Der Zusatz `.UTF-8` findet sich jedoch nicht bei den Dateien unter `/usr/share/i18n/locales` wieder, sondern zeigt nur an, dass die entsprechende Definition UTF-8-kompatibel ist.

Aus diesen Angaben generiert `locale-gen`, wie schon auf Seite 55 gesehen, die entsprechenden Lokalisierungsinformationen in `/usr/lib/locale`. Den Zusatz `.UTF-8` konvertiert `locale-gen` dabei in `utf8`, und aus diesem Grunde gibt `locale -a` die Variante `de_DE.UTF-8` als `de_DE.utf8` aus. Groß- bzw. Kleinschreibung ist hier relevant, darum müssen wir, wie oben angegeben, `LANG="de_DE.utf8"` in `/etc/env.d/02locale` setzen.

8.3.3 Benutzerspezifische Lokalisierung

Ein so konfiguriertes System sollte beim Login automatisch die deutsche Unicode-Umgebung wählen. Überprüfen lässt sich das wieder mit einem Aufruf des Befehls `locale`, diesmal ohne Argumente:

```
gentoo ~ # locale
LANG=de_DE.utf-8
LC_CTYPE="de_DE.utf-8"
LC_NUMERIC="de_DE.utf-8"
LC_TIME="de_DE.utf-8"
```

```
LC_COLLATE="de_DE.utf-8"
LC_MONETARY="de_DE.utf-8"
LC_MESSAGES="de_DE.utf-8"
LC_PAPER="de_DE.utf-8"
LC_NAME="de_DE.utf-8"
LC_ADDRESS="de_DE.utf-8"
LC_TELEPHONE="de_DE.utf-8"
LC_MEASUREMENT="de_DE.utf-8"
LC_IDENTIFICATION="de_DE.utf-8"
LC_ALL=de_DE.utf-8
```

`locale` zeigt hier nicht nur den von uns gewählten Wert für `LANG`, sondern darüber hinaus alle Unterbereiche der Lokalisierungsinformationen. Darunter finden sich z. B. die Definitionen für das Zahlenformat (`LC_NUMERIC`), das Datumsformat (`LC_TIME`) und so weiter. Mit `LANG` legen wir alle diese Sektionen auf den gleichen Wert fest, was auch der Normalsituation entspricht. Wer aber z. B. ein anderes Währungsformat bevorzugt, kann `LC_MONETARY` mit einem anderen Wert belegen.

`locale` bietet außer den hier genannten Möglichkeiten, den Befehl ohne Argumente aufzurufen, um einen Überblick über die aktuellen Einstellungen zu erhalten, und der Option `-a` zur Ausgabe der verfügbaren Locale-Definitionen noch die Option, einzelne Werte aus der derzeitigen Locale abzufragen.

Möchte man z. B. wissen, wie der Dezimaltrenner aktuell gesetzt ist, reicht der Aufruf des `locale`-Befehls mit dem Zusatz `decimal_point`:

```
gentoo ~ # locale decimal_point
,
```

In der deutschen Einstellung erhält man hier korrekterweise das Komma. Mit der Option `-k` liefert `locale` auch noch einmal den Namen des Schlüsselwortes:

```
gentoo ~ # locale -k decimal_point
decimal_point=","
```

Die Option `-c` liefert die entsprechende Kategorie:

```
gentoo ~ # locale -c -k decimal_point
LC_NUMERIC
decimal_point=","
```

Über diese Option lässt sich auch eine ganze Kategorie mit den innerhalb dieser Kategorie vorhandenen Schlüsselwörtern abfragen:

```
gentoo ~ # locale -k LC_NUMERIC
decimal_point=","
thousands_sep="."
grouping=3;3
numeric-decimal-point-wc=44
numeric-thousands-sep-wc=46
numeric-codeset="UTF-8"
```

Benutzer, die eine von der Systemeinstellung abweichende Umgebung wählen möchten, können dies über die Datei `~/.bash_profile` tun. Hier genügt es, den ursprünglichen Wert von `LANG` neu zu definieren:

```
gentoo ~ # echo 'export LANG="en_US.utf8"' >> ~/.bash_profile
```

8.3.4 Lokalisierung für Portage

Wer die Lokalisierung in der Datei `/etc/env.d/02locale` systemweit auf `LANG="de_DE.utf8"` festgelegt hat, tut dies somit auch für das Portage-System selbst und erhält manche Information, die `emerge` während des Kompilierens anzeigt, plötzlich auf Deutsch.

Das mag für den Benutzer angenehm sein; schwierig wird es aber, wenn ein Fehler auftritt und man einen Bug-Eintrag in der Gentoo-Bug-Datenbank anlegen muss: Die englischsprachigen Entwickler haben verständlicherweise gewisse Probleme mit deutschen Fehlermeldungen. Um solche Probleme zu vermeiden, sollte man Portage grundsätzlich in Englisch arbeiten lassen, indem man folgenden Eintrag mit `nano` in `/etc/portage/bashrc` vornimmt:

```
gentoo ~ # cat /etc/portage/bashrc
export LC_ALL="C"
export LANG="C"
```

Kapitel 9

Konfigurationsvariablen

In den vorangegangenen Kapiteln haben wir fast alle zentralen Konfigurationsdateien besprochen. Darüber hinaus haben einzelne Programme oder Services oft ihre eigenen, ganz spezifischen Konfigurationsoptionen, die sich in den allermeisten Fällen über entsprechende Dateien in `/etc` beeinflussen lassen.

Die Art der Konfiguration ist von Paket zu Paket unterschiedlich, so dass wir in diesem Buch nur ausgewählte Pakete beschreiben. Allerdings lassen sich grundsätzlich zwei Verfahren unterscheiden, die häufig Anwendung finden. Diese beiden Mechanismen wollen wir hier beleuchten.

Jede dieser Konfigurationsvarianten verwendet ein spezielles Verzeichnis innerhalb von `/etc`. Umgebungsvariablen können wir in `/etc/env.d` festlegen und servicespezifische Variablen in `/etc/conf.d`.

Abschließend kümmern wir uns noch kurz um die Datei `/etc/rc.conf`, die früher sehr viele Variablen enthielt. Mittlerweile sind die Entwickler dazu übergangen, die meisten auf Dateien innerhalb von `/etc/conf.d` zu verteilen, so dass nur noch wenige Änderungen in `rc.conf` notwendig sind.

9.1 Umgebungsvariablen

Viele Programme, die wir über die Kommandozeile aufrufen, beziehen so genannte Umgebungsvariablen in ihre Konfiguration ein. Gentoo definiert diese Variablen im Verzeichnis `/etc/env.d`, und wir wollen im Folgenden beschreiben, wie sich die Werte modifizieren lassen.

9.1.1 Die wichtigsten Variablen

Die aktuell definierten Umgebungsvariablen lassen sich unter `bash` mit `export` anzeigen. Folgender Befehl listet den Inhalt der Umgebungsvariable `PATH`:

```
gentoo ~ # export | grep " PATH"
declare -x PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
:/bin:/opt/bin:/usr/i686-pc-linux-gnu/gcc-bin/4.1.1"
```

Diese Variable ist von zentraler Bedeutung für die Kommandozeile, denn sie bestimmt, wo das System nach ausführbaren Programmen sucht. Generell sind unter Gentoo (und auch vielen anderen Distributionen) folgende Variablen von besonderer Bedeutung:

`PATH`
: Listet die Ordner, in denen das System nach ausführbaren Dateien sucht. Tippt man ein Kommando wie `emerge` ein, so muss eine entsprechende Datei in den im `PATH` gelisteten Verzeichnissen vorkommen, damit das System das Kommando erfolgreich ausführen kann.

`ROOTPATH`
: Hat die gleiche Funktion wie `PATH`, gilt jedoch ausschließlich für den `root`-Benutzer.

`LDPATH`
: Bezeichnet die Orte, an denen sich Bibliotheken befinden können.

`MANPATH`
: Listet die Orte, an denen sich `man`-Seiten befinden. Diese kann man dann mit dem `man`-Befehl lesen.

`INFODIR`
: Listet die Orte, an denen sich `info`-Seiten befinden. Diese kann man mit dem `info`-Befehl anzeigen.

`PAGER`
: Bezeichnet das bevorzugte Anzeige-Programm für Textdateien (üblicherweise `less`).

EDITOR
: Bezeichnet das bevorzugte Editier-Programm (üblicherweise `nano`, `vi` oder `emacs`).

CONFIG_PROTECT
: Listet die Verzeichnisse, in denen Portage während eines `emerge`-Vorgangs keine Dateien überschreiben wird, sondern dem Nutzer ermöglicht, sie mittels `etc-update` oder `dispatch-conf` zu aktualisieren (siehe auch Seite 219).

CONFIG_PROTECT_MASK
: Diese Variable maskiert innerhalb der in `CONFIG_PROTECT` angegebenen Verzeichnisse einzelne Bereiche, in denen es Portage dennoch erlaubt ist, Dateien zu überschreiben (siehe auch Seite 219).

9.1.2 Umgebungsvariablen modifizieren

Wie bereits erwähnt, dient das Verzeichnis `/etc/env.d` dazu, die Umgebungsvariablen zu definieren. Hier findet sich eine Dateiliste mit numerischen Präfixen:

```
gentoo ~ # ls /etc/env.d
00basic
02locale
05binutils
05gcc
05portage.envd
50ncurses
70klibc
70less
99gentoolkit-env
binutils
gcc
```

Die vorangestellte Zahl bestimmt den Rang einer Konfigurationsdatei. Die Dateien mit den niedrigeren Zahlen wertet `env-update` zuerst aus und überschreibt ihre Werte, wenn nachfolgende Konfigurationsdateien dieselben Variablen festlegen.

Führt man das Kommando `env-update` aus, liest dieses die einzelnen Konfigurationsdateien aus und kombiniert sie in der Datei `/etc/profile.env`.

```
gentoo ~ # cat /etc/profile.env
# THIS FILE IS AUTOMATICALLY GENERATED BY env-update.
# DO NOT EDIT THIS FILE. CHANGES TO STARTUP PROFILES
# GO INTO /etc/profile NOT /etc/profile.env

export CONFIG_PROTECT_MASK='/etc/terminfo /etc/revdep-rebuild'
```

```
export CVS_RSH='ssh'
export GCC_SPECS=''
export INFOPATH='/usr/share/info:/usr/share/binutils-data/i686-pc-linux-
gnu/2.16.1/info:/usr/share/gcc-data/i686-pc-linux-gnu/4.1.1/info'
export LANG='de_DE.utf8'
export LC_ALL='de_DE.utf8'
export LESS='-R -M --shift 5'
export LESSOPEN='|lesspipe.sh %s'
export MANPATH='/usr/local/share/man:/usr/share/man:/usr/share/binutils-
data/i686-pc-linux-gnu/2.16.1/man:/usr/share/gcc-data/i686-pc-linux-gnu/
4.1.1/man'
export PAGER='/usr/bin/less'
export PATH='/opt/bin:/usr/i686-pc-linux-gnu/gcc-bin/4.1.1'
export PRELINK_PATH_MASK='/usr/lib/klibc'
export PYTHONPATH='/usr/lib/portage/pym'
export ROOTPATH='/opt/bin:/usr/i686-pc-linux-gnu/gcc-bin/4.1.1'
```

Schaut man sich die PATH-Variable an, bemerkt man allerdings, dass die oben angegebene Regel, nach der Umgebungsvariablen entsprechend der Zahl am Dateianfang überschrieben werden, nicht ganz stimmen kann:

```
gentoo ~ #  grep "^PATH" /etc/env.d/*
/etc/env.d/00basic:PATH="/opt/bin"
/etc/env.d/05gcc:PATH="/usr/i686-pc-linux-gnu/gcc-bin/4.1.1"
```

Eigentlich müsste der Wert aus /etc/env.d/05gcc den entsprechenden PATH-Wert aus /etc/env.d/00basic überschreiben. Bei den Pfaden zu ausführbaren Dateien ist dies aber wenig sinnvoll. So installieren einige Pakete Programme in /opt/bin und der gcc-Compiler einige Werkzeuge unter /usr/i686-pc-linux-gnu/gcc-bin/4.1.1. In solchen Fällen sollte man durchaus beide Pfade in die globale PATH-Variable aufnehmen. Das entsprechende Verhalten sehen wir ja auch im Resultat:

```
export PATH='/opt/bin:/usr/i686-pc-linux-gnu/gcc-bin/4.1.1'
```

Dennoch ist das die Ausnahme für die Werte innerhalb von /etc/env.d. Diese gilt, innerhalb von Portage fest kodiert, für folgende Werte:

KDEDIRS
PATH
CLASSPATH
LDPATH
MANPATH
INFODIR
INFOPATH
ROOTPATH
CONFIG_PROTECT
CONFIG_PROTECT_MASK

```
PRELINK_PATH
PRELINK_PATH_MASK
PYTHONPATH
ADA_INCLUDE_PATH
ADA_OBJECTS_PATH
PKG_CONFIG_PATH
```

Diesen Mechanismus machen sich verschiedene Pakete zu Nutze, um die entsprechenden Variablen zu ergänzen. So lassen sich problemlos neue Pfade für ausführbare Dateien oder Hilfe-Seiten hinzufügen.

9.2 Servicevariablen

Kommen wir zum zweiten Mechanismus, der die Konfiguration für die Service-Skripte unter `/etc/init.d` vereinheitlicht. Die entsprechenden Dateien befinden sich unter `/etc/conf.d`.

9.2.1 Zuordnung

Die Konfigurationsdateien für ein spezifisches Skript sind im Normalfall analog benannt. Also ist `/etc/conf.d/apache2` für die Konfiguration von `/etc/init.d/apache2` zuständig. Nicht jedes `init.d`-Skript hat jedoch zwangsläufig eine Konfigurationsdatei.

Auch befindet sich häufig nicht die komplette Konfiguration für den Service in der Datei unter `/etc/conf.d`. Die Apache-Konfiguration z. B. ist sehr komplex, und der Großteil der Konfigurationsdateien befindet sich unter `/etc/apache2`, während `/etc/conf.d/apache2` nur einige Basis-Einstellungen erlaubt.

Einige der Konfigurationsdateien haben wir uns schon in anderen Kapiteln angesehen (`hostname` auf Seite 51, `keymaps` auf Seite 54 und Seite 183, `consolefont` auf Seite 184, `clock` auf Seite 180 und `net` auf Seite 80). Die übrigen, allgemeinen Dateien beschreiben wir hier.

9.2.2 /etc/conf.d/rc

Diese Konfigurationsdatei definiert nicht die Variablen für ein spezifisches Init-Skript, sondern liefert die Grundkonfiguration für alle Init-Skripte. Über diese Datei lassen sich somit einige zentrale Eigenschaften des Boot-Vorgangs steuern.

Die Datei ist etwas zu lang, um sie hier in Gänze darzustellen – darum hier nur der Überblick über die Liste der verfügbaren Variablen und deren Standardwerte:

```
RC_TTY_NUMBER=11
RC_PARALLEL_STARTUP="no"
RC_INTERACTIVE="yes"
RC_HOTPLUG="yes"
RC_COLDPLUG="yes"
RC_PLUG_SERVICES=""
RC_NET_STRICT_CHECKING="no"
RC_DOWN_INTERFACE="yes"
RC_VOLUME_ORDER="raid evms lvm dm"
RC_VERBOSE="no"
RC_BOOTLOG="no"
RC_BOOTCHART="no"
RC_USE_FSTAB="no"
RC_USE_CONFIG_PROFILE="yes"
RC_FORCE_AUTO="no"
RC_DEVICES="auto"
RC_DEVICE_TARBALL="no"
RC_SWAP_ERASE="no"
RC_DMESG_LEVEL="1"
RC_RETRY_KILL="yes"
RC_RETRY_TIMEOUT=1
RC_RETRY_COUNT=5
RC_FAIL_ON_ZOMBIE="no"
RC_KILL_CHILDREN="no"
RC_WAIT_ON_START="0.1"
#RC_DAEMON="/usr/bin/valgrind --tool=memcheck --log-file=/tmp/valgrind.s
yslog-ng"
```

`RC_TTY_NUMBER`

Die Variable `RC_TTY_NUMBER` bestimmt die Zahl an Terminals (*ttys*), die beim Systemstart geöffnet werden. Der Standardwert ist auf elf gesetzt, aber bei einem Server-System kann man den Wert problemlos reduzieren. Selbst für eine Desktop-Maschine ist der Wert hoch, da die meisten Benutzer nur den X-Server starten und die anderen Terminals unbenutzt bleiben. Die Ressourcen, die man durch eine Reduktion des Wertes spart, sind allerdings minimal.

`RC_PARALLEL_STARTUP`

Die Option `RC_PARALLEL_STARTUP` ist da schon spannender, da sie bestimmt, ob das System während des Boot-Vorgangs die Init-Skripte parallel starten soll oder nicht. Die eingesparte Zeit ist zwar nicht groß, aber wer wartet schon gern. Ganz ohne Vorsicht ist die Einstellung nicht zu genießen: Gentoo unterstützt das Verfahren noch nicht lange, und aus diesem Grunde ist der Standardwert der Variablen auf `no` gesetzt. Nach Aktivieren der Option sollte man beim nächsten Boot-Vorgang nach Problemen Ausschau halten.

Beim parallelen Boot-Vorgang wird die Ausgabe der Init-Skripte gekürzt, damit sich die Ausgaben verschiedener Skripte nicht überlagern und damit nicht mehr zuzuordnen sind.

RC_INTERACTIVE
: RC_INTERACTIVE="yes" erlaubt es, während des Systemstarts die Taste [I] zu verwenden, um den Boot-Vorgang zu unterbrechen und bei jedem Service zu wählen, ob er gestartet werden soll oder nicht. Vor allem bei Boot-Problemen ist diese Möglichkeit sehr nützlich.

RC_*PLUG*
: RC_HOTPLUG legt fest, ob die Init-Skripte auch durch ein Hotplug-Event angestoßen werden dürfen. Das kann vor allem bei mobilen Maschinen oder USB-Hardware sinnvoll sein. Fügt man neue Netzwerkkarten oder WLAN-Empfänger hinzu, müssen vielfach auch einige Netzwerkdienste neu gestartet werden. Wer dies verhindern möchte, setzt abweichend von der Standardeinstellung RC_HOTPLUG="no".

  Die Option RC_COLDPLUG hat den gleichen Zweck und verhindert, wenn gewünscht, den Aufruf von Init-Skripten in der Coldplug-Phase während des Systemstarts. Darüber hinaus kann man mit der Einstellung RC_PLUG_SERVICES genauer spezifizieren, welche Services bei einem Cold- bzw. Hotplug-Ereignis gestartet werden dürfen. Die Syntax ist in den Kommentaren der Datei /etc/conf.d/rc beschrieben.

RC_NET_STRICT_CHECKING
: Eine Vielzahl von Services hängt von einem funktionierenden Netzwerk ab (siehe auch Kapitel 7.3.1 ab Seite 174) und lässt sich erst starten, wenn dieses verfügbar ist. Viele Rechner besitzen aber mittlerweile mehrere Netzwerkschnittstellen, und die Bedingung „Netzwerk verfügbar“ lässt sich unterschiedlich formulieren. Über den Parameter RC_NET_STRICT_CHECKING lässt sich bestimmen, wie dies geschieht.

  Bei RC_NET_STRICT_CHECKING="none" geht das Init.d-System stets davon aus, dass ein funktionierendes Netzwerk zur Verfügung steht. Mit den beiden Einstellung RC_NET_STRICT_CHECKING="no" bzw. RC_NET_STRICT_CHECKING="lo" muss für diese Annahme mindestens ein Netzwerkskript ausschließlich bzw. einschließlich net.lo erfolgreich gestartet sein. Die Einstellung "lo" unterscheidet sich in der Praxis nicht von "none", da jede normal konfigurierte Maschine das Loopback-Interface startet. Schließlich verlangt die Option RC_NET_STRICT_CHECKING="yes", dass das System *alle* Schnittstellen erfolgreich starten muss, damit es das Netzwerk als etabliert betrachten darf.

RC_DOWN_INTERFACE
: RC_DOWN_INTERFACE legt fest, ob das System die Netzwerkschnittstellen beim Herunterfahren vollständig abschalten soll. Wer seine Maschine mit Wake-on-LAN betreibt muss hier die Einstellung no wählen.

RC_VOLUME_ORDER
: Die Einstellung RC_VOLUME_ORDER erlaubt für Maschinen mit komplexer Festplattenkonfiguration und verschiedenen Festplatten-Management-Systemen (RAID, LVM etc.) die Startabfolge dieser Systeme festzulegen.

RC_VERBOSE
: Wer beim Booten Probleme erkennt oder Services sieht, die nicht erfolgreich starten, dem kann RC_VERBOSE="yes" vielleicht helfen. Die Option soll die Menge der von den Init-Skripten zurückgegebenen Informationen erhöhen. Bisher respektieren aber nur wenige Init.d-Skripte diese Einstellung. Ist jedoch RC_PARALLEL_STARTUP="yes" gesetzt (d. h. die Meldungen fehlen völlig), bekommt man mit dieser Option die Ausgabe der Init-Skripte zurück. In diesem Fall können sich aber die Ausgaben verschiedener Init-Skripte überlappen.

RC_BOOTLOG
: Ebenfalls für das Debuggen von Problemen beim Boot-Vorgang nützlich ist die Option RC_BOOTLOG, die es mit dem Wert yes erlaubt, die Ausgabe des Boot-Vorgangs in der Datei /var/log/boot.msg abzuspeichern. Vor allem bei Maschinen ohne Monitor kann das sehr hilfreich sein. Dafür muss das Paket app-admin/showconsole installiert sein, außerdem sollte der Boot-Vorgang auf die Darstellung eines Splash-Screens verzichten, da showconsole sonst versagt.

RC_BOOTCHART
: Gerade bei Desktop-Maschinen sind viele Nutzer an einem möglichst schnellen Start der Maschine interessiert. Um hier Optimierungsmöglichkeiten zu identifizieren, kann man das bootchart-Programm verwenden (app-benchmarks/bootchart), das den zeitlichen Verlauf beim Booten grafisch veranschaulicht und damit Flaschenhälse während des Startprozesses auffindbar macht. RC_BOOTCHART="yes" aktiviert dann das Profiling des Boot-Vorgangs. Wie man anschließend aus den gesammelten Daten einen grafischen Überblick erstellt, beschreibt die Dokumentation des bootchart-Programms.

RC_USE_FSTAB
: Die besonderen Dateisysteme proc, sysfs und devpts bindet man normalerweise über die Verzeichnisse /proc, /sys und /dev/pts in das System ein. Sollte die /etc/fstab etwas anderes verlangen, ignoriert das Init-System dies. Über die Einstellung RC_USE_FSTAB="yes" kann man es dennoch anweisen, die Zielverzeichnisse für diese speziellen Dateisysteme aus den Einträgen der /etc/fstab-Datei zu entnehmen (siehe auch Kapitel 16.4.1 ab Seite 352).

RC_USE_CONFIG_PROFILE
: Um die Konfigurierbarkeit bei mobilen Rechnern zu erhöhen, gibt

es bei Gentoo die Möglichkeit, die Konfiguration des Init-System an den jeweiligen Runlevel anzupassen. Im Kapitel 7.1 ab Seite 167 haben wir schon besprochen, wie sich der Runlevel des Systems beim Booten über die Option `softlevel` wählen lässt. Um gegebenenfalls auch die Konfiguration eines Init.d-Skriptes an den aktuell gewählten Runlevel anzupassen, speichert man die für diese Umgebung spezifische Konfiguration in einer Datei, die am Ende den Namen des gewählten Softlevels, getrennt durch einen Punkt, trägt.

Angenommen, unter `/etc/runlevels` liegt ein Ordner `unterwegs`, der die Konfiguration für den mobilen Einsatz des Rechners festlegt, dann könnten wir hier z. B. einen geänderten Hostnamen in der Datei `/etc/conf.d/hostname.unterwegs` festlegen, während wir den Standardwert in der normalen Datei `/etc/conf.d/hostname` festlegen. Dieses Verhalten ist normalerweise erwünscht, lässt sich aber bei Bedarf über die Einstellung `RC_USE_CONFIG_PROFILE` deaktivieren.

`RC_FORCE_AUTO`
: Die Option `RC_FORCE_AUTO` vom Standardwert `no` auf `yes` zu setzen führt dazu, dass das Init.d-System beim Boot oder Stopp der Maschine versucht, jegliche Nutzer-Interaktion zu unterdrücken. Im Normalfall fordert das System beim Boot-Vorgang eine solche Interaktion nur, wenn es zu einem schwerwiegenden Fehler kommt, und darum sollte man diese Option nicht aktivieren. Bei Systemen ohne Monitor besteht jedoch im Normalfall keine Möglichkeit, mit dem System während des Hoch- bzw. Herunterfahrens zu interagieren, so dass hier der Wert `yes` sinnvoll ist.

`RC_DEVICE*`
: Die Variablen `RC_DEVICES` und `RC_DEVICE_TARBALL` beschreiben wir im Kapitel 16.4 zu `udev` ab Seite 351.

`RC_SWAP_ERASE`
: Die Variable `RC_SWAP_ERASE` ist in der `/etc/conf.d/rc` nett kommentiert und, wie dem Kommentar zu entnehmen, für paranoide Linux-Nutzer gedacht. Setzt der Benutzer die Variable auf `yes`, so wird das System den Swap-Speicher beim Herunterfahren löschen. Erfahrene Hacker könnten aus den Swap-Daten möglicherweise private Daten des Nutzers rekonstruieren, und wer meint, sich vor solchen Attacken schützen zu müssen, kann die Option entsprechend aktivieren.

`RC_DMESG_LEVEL`
: Mit der Variablen `RC_DMESG_LEVEL` kann man die Menge der Kernel-Nachrichten beeinflussen. Die Standardeinstellung ist `1`; in diesem Fall gibt der Kernel nur wirklich fatale Fehler auf der Konsole (also dem Start-Bildschirm) aus. Wer Probleme mit seiner Hardware hat,

kann den Wert erhöhen und erhält vielleicht bessere Hinweise auf mögliche Ursachen.

RC_RETRY_*, RC_FAIL_ON_ZOMBIE, RC_KILL_CHILDREN und RC_WAIT_ON_START
Bleiben noch ein paar Variablen, die das Verhalten beim Starten bzw. Stoppen von Services durch das Init.d-System beeinflussen. Diese gelten jedoch nur für die Skripte in /etc/init.d, die zum Starten und Stoppen von Services das start-stop-daemon-Programm verwenden. Dies betrifft z. B. den System-Logger syslog-ng:

```
...
start() {
  checkconfig || return 1
  ebegin "Starting syslog-ng"
  [[ -n ${SYSLOG_NG_OPTS} ]] && SYSLOG_NG_OPTS="--${SYSLOG_NG
_OPTS}"
  start-stop-daemon --start --quiet --exec /usr/sbin/syslog-ng${SY
SLOG_NG_OPTS}
   eend $? "Failed to start syslog-ng"
}

stop() {
  ebegin "Stopping syslog-ng"
  start-stop-daemon --stop --quiet --pidfile /var/run/syslog-ng.pid
  eend $? "Failed to stop syslog-ng"
  sleep 1 # needed for syslog-ng to stop in case we're restarting
}
...
```

Dabei kann man über die Option RC_RETRY_KILL="yes" festlegen, dass der start-stop-daemon mehrfach versuchen soll, den Service mit einem KILL-Signal zu stoppen, sofern dies im ersten Versuch nicht gelingt. In diesem Fall legt der Parameter RC_RETRY_TIMEOUT die Zeitspanne zwischen zwei Versuchen fest und RC_RETRY_COUNT bestimmt die Anzahl der Wiederholungen.

Angenommen das Init.d-System ist der Meinung, ein Service sei erfolgreich gestartet worden und habe noch kein Stopp-Signal vom Benutzer erhalten, so sollte beim Stoppen des Service der Prozess auch noch detektierbar sein. Ist er es nicht, beschwert sich das Init.d-System im Normalfall nicht. Wer möchte, kann in dieser Situation aber auch einen Fehler erzwingen, indem er RC_FAIL_ON_ZOMBIE auf yes setzt.

Die Option RC_WAIT_ON_START legt fest, wie lange das Init-System nach dem Start eines Service wartet, bis es überprüft, ob der Start erfolgreich war.

Zu guter Letzt kann man mit der Option RC_KILL_CHILDREN festlegen, ob das Init.d-System beim Stoppen eines Service auch all des-

sen Kind-Prozesse beendet. Das ist nicht unbedingt als globale Einstellung zu empfehlen, da in diesem Fall z. B. beim SSH-Server alle geöffneten Terminals ebenfalls automatisch geschlossen würden. Im Normalfall kann man sich auf dem Server über SSH einloggen und den `sshd`-Daemon neu starten, ohne dass dabei die eigene Verbindung zusammenbricht. Vor allem wenn man die Konfiguration des `sshd` verändert hat, ist die eine bestehende Verbindung sehr nützlich, sollte man sich vielleicht plötzlich mit der geänderten Konfiguration nicht mehr einloggen können. Die bestehende Verbindung kann man dann nutzen, um die Konfiguration zurückzusetzen.

Ähnlich sollte auch bei anderen Services eventuell laufenden Kind-Prozessen die Möglichkeit gegeben werden ihre Arbeit abzuschließen, bevor sie vollständig stoppen.

Der Rest von /etc/conf.d/rc

Es gibt noch einige weitere Variablen in dieser zentralen Konfigurationsdatei, die wir an dieser Stelle jedoch nicht weiter besprechen wollen, da sie derart tief in das Init.d-System eingreifen, dass man sehr genau wissen sollte, was man eigentlich verändert. Dafür sollte man sich intensiver mit der Dokumentation und dem Code des `sys-apps/baselayout`-Pakets auseinander setzen.

9.2.3 /etc/conf.d/bootmisc

Einige kleinere Aufgaben, die beim Boot-Vorgang anfallen, erledigt das Init-Skript `/etc/init.d/bootmisc`. Auch hier gibt es eine Konfigurationsdatei:

```
gentoo ~ # cat /etc/conf.d/bootmisc
# /etc/conf.d/bootmisc

# Put a nologin file in /etc to prevent people from logging in before
# system startup is complete

DELAYLOGIN="no"

# Should we completely wipe out /tmp or just selectively remove known
# locks / files / etc... ?

WIPE_TMP="no"
```

Für einen Server kann man `DELAYLOGIN` auf `yes` setzen, damit sich externe Benutzer erst nach dem vollständigen Boot-Vorgang einloggen können. Außerdem ist es möglich, das `/tmp`-Verzeichnis während des Boot-Vorgangs automatisch leeren zu lassen, indem man `WIPE_TMP` auf `yes` setzt.

9.2.4 /etc/conf.d/hdparm

Das Werkzeug `hdparm` ist essentiell, um den Festplattenzugriff unter Linux zu optimieren. Es ist an dieser Stelle nicht möglich, detailliert auf `hdparm` einzugehen, da die Zahl an verfügbaren Optionen recht hoch ist.

Es gibt ein Init-Skript gleichen Namens (`/etc/init.d/hdparm`), das beim Boot-Vorgang plattenspezifische Einstellungen aktiviert. Die dafür notwendige Konfiguration findet sich erwartungsgemäß in `/etc/conf.d/hdparm`:

```
gentoo ~ # cat /etc/conf.d/hdparm
# /etc/conf.d/hdparm: config file for /etc/init.d/hdparm

# You can either set hdparm arguments for each drive using hdX_args,
# discX_args, cdromX_args and genericX_args, e.g.
#
# hda_args="-d1 -X66"
# disc1_args="-d1"
# cdrom0_args="-d1"

# or you can set options for all PATA drives
pata_all_args="-d1"

# or you can set options for all SATA drives
sata_all_args=""

# or, you can set hdparm options for all drives
all_args=""
```

Allgemeine Einstellungen, die für alle Platten gelten, können wir mit Hilfe von `all_args` festlegen. Für bestimmte Plattentypen (paralleles ATA – PATA; serielles ATA – SATA; ATA = Advanced Technology Attachment) gibt es zwei Optionen (`pata_all_args`, `sata_all_args`), die global Parameter für diesen Plattentyp festlegen. Die meisten handelsüblichen Laufwerke fallen in die PATA-Klasse.

In den meisten Fällen kann man z. B. die Option `-d1` verwenden. Damit wird der DMA-Zugriff auf die Platte oder das DVD-ROM-Laufwerk aktiviert. Da die meisten modernen Laufwerke diese Option unterstützen, die den Datenfluss merklich beschleunigt, kann man sie relativ sicher eintragen. Entsprechend findet sich in der Standardkonfiguration der Eintrag `pata_all_args="-d1"` wieder.

Bevor man speziellere Einstellungen festlegt, sollte man wenn möglich mit Hilfe von `hdparm` überprüfen, ob die eigene Festplatte die gewünschten Optionen überhaupt unterstützt. Meist kann man auf die Fähigkeiten von `hdparm` vertrauen, die korrekten Einstellungen eigenständig festzustellen.

Die aktuell gesetzten Optionen kann man sich mit Hilfe des Befehls `hdparm DEVICE` ansehen. Für `/dev/hda` sieht das dann z. B. so aus:

```
gentoo ~ # hdparm /dev/hda
/dev/hda:
 multcount    = 16 (on)
 IO_support   =  1 (32-bit)
 unmaskirq    =  1 (on)
 using_dma    =  1 (on)
 keepsettings =  0 (off)
 readonly     =  0 (off)
 readahead    = 256 (on)
 geometry     = 65535/16/63, sectors = 156301488, start = 0
```

Optionen wie DMA, 32-Bit IO, unmaskierter IRQ etc. sind hier korrekt gesetzt. In den meisten Fällen ist keine Modifikation der Standardwerte in `/etc/conf.d/hdparm` notwendig. Genauere Informationen über die eigene Platte liefert `hdparm -i` *DEVICE*:

```
gentoo ~ # hdparm -i /dev/hda

/dev/hda:

 Model=SAMSUNG SP1614N, FwRev=TM100-30, SerialNo=S016J10XC24795
 Config={ HardSect NotMFM HdSw>15uSec Fixed DTR>10Mbs }
 RawCHS=16383/16/63, TrkSize=34902, SectSize=554, ECCbytes=4
 BuffType=DualPortCache, BuffSize=8192kB, MaxMultSect=16, MultSect=16
 CurCHS=4047/16/255, CurSects=16511760, LBA=yes, LBAsects=268435455
 IORDY=on/off, tPIO={min:240,w/IORDY:120}, tDMA={min:120,rec:120}
 PIO modes:  pio0 pio1 pio2 pio3 pio4
 DMA modes:  mdma0 mdma1 mdma2
 UDMA modes: udma0 udma1 *udma2 udma3 udma4 udma5
 AdvancedPM=no WriteCache=enabled
 Drive conforms to: (null):  ATA/ATAPI-1 ATA/ATAPI-2 ATA/ATAPI-3 ATA/ATA
PI-4 ATA/ATAPI-5 ATA/ATAPI-6 ATA/ATAPI-7

 * signifies the current active mode
```

Beide oben genannten Befehle dürften einen Überblick geben, ob die Platte korrekt funktioniert. Einen Testlauf führt man mit `hdparm -tT DEVICE` durch.

```
gentoo ~ # hdparm -tT /dev/hda
/dev/hda:
 Timing cached reads:   924 MB in  2.00 seconds = 461.21 MB/sec
 Timing buffered disk reads:   74 MB in  3.08 seconds =  24.01 MB/sec
```

Dieser Testlauf lässt sich nutzen, um die Performance der Platte bei Veränderungen an der Konfiguration zu überprüfen. Solche Spielereien sollte man allerdings nur vornehmen, wenn man sich sicher ist, dass die automatische Konfiguration nicht vernünftig arbeitet.

9.2.5 /etc/conf.d/local.*

Um das Verhalten beim Starten bzw. Herunterfahren des Systems zu modifizieren, kann man in den Dateien `/etc/conf.d/local.start` und `/etc/conf.d/local.stop` eigene Befehle einfügen. Das zugehörige Init-Skript heißt in diesem Fall allerdings nur `/etc/init.d/local`.

```
gentoo ~ # cat /etc/conf.d/local.start
# /etc/conf.d/local.start

# This is a good place to load any misc programs
# on startup (use &>/dev/null to hide output)

gentoo ~ # cat /etc/conf.d/local.stop
# /etc/conf.d/local.stop

# This is a good place to unload any misc.
# programs you started above.
# For example, if you are using OSS and have
# "/usr/local/bin/soundon" above, put
# "/usr/local/bin/soundoff" here.
```

`local.start` benutzt der Autor z. B., um über das Programm `setkeycodes` beim Hochfahren des Systems einige besondere Tasten auf der Tastatur im Kernel zu deklarieren:

```
gentoo ~ # setkeycodes e016 174 e064 212 e03c 213
```

9.2.6 /etc/rc.conf

Nahezu alle Konfigurationsvariablen, die früher einmal in dieser Datei zu finden waren, sind in andere Dateien innerhalb von `/etc/conf.d` gewandert. Übrig blieben `UNICODE`, `EDITOR` und `XSESSION`.

`UNICODE` haben wir schon kurz während der Installation auf Seite 54 erwähnt. Da wir hier ein Unicode-System aufbauen und die Variable standardmäßig auf `yes` gesetzt ist, besteht hier kein Änderungsbedarf.

Auch `EDITOR` fand schon bei der Installation in Kapitel 1.15.1 Erwähnung. Hier sollte jeder einfach seinen bevorzugten Editor wählen.

Und schließlich dient `XSESSION` dazu, den Fenstermanager für den X-Server auszuwählen. Da wir an dieser Stelle keine grafische Benutzeroberfläche installiert haben, lassen wir die Variable auch weiter auskommentiert.

Kapitel

10 Gentoo frisch halten

Wir haben schon in der Einleitung erwähnt, dass Gentoo dem Prinzip „Install once, never reinstall“ folgt. Sofern wir unser System also nicht aus Versehen massiv beschädigen, sollten wir die Prozedur in Kapitel 1 *nie* wiederholen müssen. Das System unter Murren neu zu installieren, weil „irgendwie“ gar nichts mehr geht, gehört damit der Vergangenheit an.

Es wäre aber nicht ganz richtig zu behaupten, Aktualisierungen unter Gentoo seien mit keinerlei Aufwand verbunden; diese erfolgen aber nicht im Rahmen einer kompletten Neuinstallation.

Gentoo besitzt mit dem Portage-Baum ein sehr lebendiges Archiv an Paketinformationen, das die Entwickler konstant aktualisieren. Es gibt also im eigentlichen Sinne keinen „Versionssprung“ von einem Gentoo-Release zum nächsten.[1] Damit lassen sich Aktualisierungen, die von den meisten anderen Distributionen bzw. Betriebssystemen im Rahmen eines Versions-

[1] Die Installations-CDs/DVDs mit Versionsnummern wie 2007.0 oder 2008.0 werden lediglich zu einem beliebigen Zeitpunkt vom Portage-Baum generiert, besonders gut getestet und dann den Benutzern zur Verfügung gestellt. Sie stellen darüber hinaus aber keinen Meilenstein des Portage-Baums dar.

sprungs zusammengepackt werden, in kleinere Portionen aufteilen und in kleineren Schritten durchführen. Vorschläge zu einem geeigneten Aktualisierungsschema geben wir am Ende des Kapitels ab Seite 241.

Bringen wir nun das System auf den neuesten Stand, entsteht bei der Lektüre dieses Buches folgendes Problem: Einige Beispiele werden von den Gegebenheiten auf Ihrem System abweichen; auch wenn es sich meist nur um veränderte Versionsnummern handelt, ist nicht auszuschließen, dass das ein oder andere Beispiel nicht mehr vollständig zu reproduzieren ist.

Wenn Sie sich bis in dieses Kapitel vorgearbeitet haben, sind Sie zwar auch kein Gentoo-Anfänger mehr und werden vermutlich nicht verzweifeln; dennoch möchten wir Ihnen die Möglichkeit geben (und auch empfehlen), die Beispiele wie hier beschrieben nachzuvollziehen. Darum die Empfehlung, die nächsten beiden Abschnitte 10.1 und 10.2 zunächst einmal *nur zu lesen* und die Kommandos noch nicht auszuführen.

Die Beispiele ab 10.3 basieren dann wieder auf dem System der beiliegenden LiveDVD. Sobald Sie sämtliche Themen dieses Buches, die Sie interessieren, bearbeitet haben, kehren Sie an diese Stelle zurück und aktualisieren Ihr System mit `emerge --sync`, gefolgt von `emerge -uND world`.

10.1 Portage-Baum aktualisieren

Das Aktualisieren einer Gentoo-Distribution beginnt, wie auch schon während der Installation kurz gesehen (siehe 1.6.1), grundsätzlich mit

```
gentoo ~ # emerge --sync
>>> Starting rsync with rsync://130.230.54.100/gentoo-portage...
>>> Checking server timestamp ...
...
```

Wie auch schon auf Seite 40 erwähnt, lassen sich die ausführlichen `rsync`-Informationen mit der Option `--quiet` (bzw. `-q`) unterdrücken.

Hiermit bringen wir den Portage-Baum auf den neuesten Stand und synchronisieren ihn mit einem der Mirror-Server. Sollte der Befehl mit der Nachricht enden, dass eine neue Version von `sys-apps/portage` verfügbar ist (siehe Kapitel 1.6.1), sollte der erste Befehl nach dem Synchronisieren lauten:

```
gentoo ~ # emerge -av sys-apps/portage

These are the packages that would be merged, in order:

Calculating dependencies... done!

[ebuild     U ] app-shells/bash-3.2_p17-r1 [3.1_p17] USE="nls -afs -bash
```

```
logger -plugins -vanilla" 2,522 kB
[ebuild     U ] sys-apps/sandbox-1.2.18.1-r2 [1.2.17] 232 kB
[ebuild     U ] sys-apps/portage-2.1.3.19 [2.1.2.2] USE="-build -doc -ep
ydoc (-selinux)" LINGUAS="-pl" 387 kB
*** Portage will stop merging at this point and reload itself,
    then resume the merge.
[ebuild     U ] dev-python/pycrypto-2.0.1-r6 [2.0.1-r5] USE="-bindist -g
mp -test" 0 kB

Total: 4 packages (4 upgrades), Size of downloads: 3,139 kB

Would you like to merge these packages? [Yes/No] Yes
```

Der Portage-Baum liefert die Basisdaten für die Arbeit von `emerge` und die anderen Werkzeuge, die in `sys-apps/portage` enthalten sind. Entsprechend ist dieses Update nach einer Synchronisation so wichtig, um sicher zu gehen, dass die Interaktion der beiden Komponenten reibungslos funktioniert.

Es kann auch passieren, dass `emerge --sync` am Ende der Synchronisation noch folgende Nachricht ausspuckt:

```
 * IMPORTANT: 3 config files in '/etc' need updating.
 * Type emerge --help config to learn how to update config files.
```

Wir behandeln die Aktualisierung von Konfigurationsdateien nach einem Paket-Update erst wieder in Abschnitt 10.3. Das Besondere an der oben gezeigten Nachricht ist aber, dass sie nicht nach einem Paket-Update erscheint, sondern nach der Synchronisation des Portage-Baums.

Die Meldung besagt, dass es Dateien in `/etc` zu aktualisieren gibt, obwohl wir eigentlich nur die Dateien in `/usr/portage` aktualisieren wollten.

Die zu aktualisierenden Dateien, die von `emerge --sync` betroffen sein können, liegen alle in `/etc/portage` und sind die `package.*`-Dateien, die wir in Kapitel 5 recht ausführlich beleuchtet haben.

Wenn Sie z. B. den Installationsangaben dieses Buches Schritt für Schritt gefolgt sind, dann haben Sie in `/etc/portage/package.use` z. B. die Zeile `net-www/apache mpm-prefork mpm-worker threads` stehen.

Nun ist das Paket `net-www/apache` aber in der Zwischenzeit aus der Kategorie `net-www` in `www-servers` gewandert. Wir sehen also, dass die Paketkategorien nicht fix sind. Die Entwickler achten darauf, dass einzelne Kategorien nicht zu groß oder zu klein werden und eine gewisse Übersichtlichkeit erhalten bleibt.

Für unsere Datei `/etc/portage/package.use` bedeutet das aber in diesem Fall, dass die Angabe `net-www/apache` nicht mehr korrekt ist und wir beim nächsten Apache-Upgrade unsere Einstellungen der USE-Flags verlieren würden.

Portage ist deshalb in der Lage, nach einem `emerge --sync` alle Probleme zu beseitigen, die durch einen Wechsel der Kategorie entstehen können. Das betrifft nicht nur die Dateien in `/etc/portage`, aber das ist die einzige Aktion, die einen direkten Einfluss auf den Benutzer hat.

Wir greifen also hier einmal auf `dispatch-conf` (siehe Seite 219) vor und aktualisieren unsere Konfiguration in `/etc/portage`, indem wir das Programm aufrufen und drei Mal mit `[U]` die Aktualisierung akzeptieren:

```
--- /etc/portage/package.keywords       2008-01-25 19:46:31.000000000 +0
100
+++ /etc/portage/._cfg0000_package.keywords     2008-01-28 09:59:26.0000
00000 +0100
@@ -1,3 +1,3 @@
-net-www/apache ~x86
-=net-www/apache-2.0.59-r2 ~x86
+www-servers/apache ~x86
+=www-servers/apache-2.0.59-r2 ~x86
 =dev-libs/apr-util-1.2.8 ~x86

>> (1 of 3) -- /etc/portage/package.keywords
>> q quit, h help, n next, e edit-new, z zap-new, u use-new
   m merge, t toggle-merge, l look-merge:  u
--- /etc/portage/package.unmask 2008-01-25 19:39:55.000000000 +0100
+++ /etc/portage/._cfg0000_package.unmask       2008-01-28 09:59:26.0000
00000 +0100
@@ -1 +1 @@
->=net-www/apache-2.2.0
+>=www-servers/apache-2.2.0

>> (2 of 3) -- /etc/portage/package.unmask
>> q quit, h help, n next, e edit-new, z zap-new, u use-new
   m merge, t toggle-merge, l look-merge:  u
--- /etc/portage/package.use    2008-01-25 19:15:14.000000000 +0100
+++ /etc/portage/._cfg0000_package.use  2008-01-28 09:59:26.000000000 +0
100
@@ -1 +1 @@
-net-www/apache mpm-prefork mpm-worker threads
+www-servers/apache mpm-prefork mpm-worker threads

>> (3 of 3) -- /etc/portage/package.use
>> q quit, h help, n next, e edit-new, z zap-new, u use-new
   m merge, t toggle-merge, l look-merge:  u
```

10.1.1 Verbindungsprobleme beim Aktualisieren

In manchen Netzwerkumgebungen wird eine Firewall die Synchronisation mit Hilfe von `rsync` über Port 873 verhindern. In diesem Fall ist es möglich, auf HTTP über Port 80 auszuweichen.

Das Skript `emerge-webrsync` lädt ein Snapshot-Archiv des Portage-Baums über HTTP herunter und aktualisiert darüber unsere lokale Kopie des Baums. Es ersetzt somit `emerge --sync`:

```
gentoo ~ # emerge-webrsync
Fetching most recent snapshot
Attempting to fetch file dated: 20080127
portage-20080127.tar.bz2: OK
Syncing local tree...

...

>>> Updating Portage cache:  100%

 *** Completed websync, please now perform a normal rsync if possible.
     Update is current as of the of YYYYMMDD: 20080127
```

Der Nachteil dieses Vorgehens: Wir laden den kompletten Portage-Baum herunter, und das sind immerhin 40 MB Daten, von denen der größte Teil eigentlich schon auf dem eigenen System existiert.

Wenn `rsync` konstant durch eine Firewall blockiert ist, wäre es also deutlich effizienter, nur die Unterschiede zum letzten Synchronisationsstand herunterzuladen. Das erlaubt das Paket `app-portage/emerge-delta-webrsync`, das wir allerdings, wenn gewünscht, gesondert installieren müssen.

```
gentoo ~ # emerge -av app-portage/emerge-delta-webrsync

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] dev-util/diffball-0.7.1  USE="-debug" 325 kB
[ebuild  N    ] app-arch/tarsync-0.2.1  14 kB
[ebuild  N    ] app-portage/emerge-delta-webrsync-3.5.1-r2  13 kB

Total: 3 packages (3 new), Size of downloads: 351 kB

Would you like to merge these packages? [Yes/No]
gentoo ~ # emerge-delta-webrsync
Looking for available base versions for a delta
fetching patches
...
```

Vor allem für Systeme mit geringer Bandbreite ist dies ein nützliches Werkzeug.

Zum Abschluss der Synchronisation führt Portage automatisch ein Update des so genannten *Metadaten-Cache* aus. Dieser speichert unter `/usr/portage/metadata/cache` einige grundlegende Daten der Ebuilds wie z. B. die Homepage, USE-Flags, Keywords usw. in einem Format, auf das `emerge` schnell zugreifen kann.

```
gentoo ~ # cat /usr/portage/metadata/cache/sys-libs/zlib-1.2.3-r1
...
http://www.gzip.org/zlib/zlib-1.2.3.tar.bz2

http://www.zlib.net/
ZLIB
Standard (de)compression library
alpha amd64 arm hppa ia64 m68k mips ppc ppc64 s390 sh sparc ~sparc-fbsd
x86 ~x86-fbsd
multilib toolchain-funcs eutils portability flag-o-matic
...
```

Den Cache können wir auch losgelöst vom Befehl `emerge --sync` über `emerge --metadata` aktualisieren. Im Normalfall ist dies allerdings nicht notwendig.

10.2 Pakete aktualisieren

Nach der Erstinstallation liegen nicht zwingend alle Pakete in der neuesten Version vor, schließlich sind wir von einer vorkompilierten Stage ausgegangen, deren Erstellungsdatum wahrscheinlich schon einige Zeit zurück liegt. Entsprechend sind für viele Pakete schon neuere Versionen verfügbar, da sich der Portage-Baum kontinuierlich weiter entwickelt.

Wir wollen veraltete Pakete erneuern und wissen bereits, dass wir die neueste Apache-Version mit `emerge www-servers/apache` installieren können. Der Befehl trifft allerdings keine Unterscheidung, ob das zu installierende Paket schon in der neuesten Version vorliegt oder nicht. Angenommen es gibt keine neuere Version als die derzeit installierte, verschwenden wir einige Zeit damit, das schon vorhandene Paket nochmals zu kompilieren und zu installieren.

Damit ein Versionsunterschied ausschlaggebend für die Installation eines Pakets ist, fügen wir dem `emerge`-Aufruf die Option `--update` (bzw. `-u`) hinzu. Damit führt `emerge` nur dann eine Aktion aus, wenn auch wirklich eine neuere Paketversion verfügbar ist. Da wir an dieser Stelle das Paket noch nicht wirklich aktualisieren wollen, sondern erst einmal überprüfen möchten, was Portage überhaupt machen würde, fügen wir die schon aus Kapitel 4.2.1 bekannte Option `--pretend` (bzw. `-p`) ein.

```
gentoo ~ # emerge -pu www-servers/apache

These are the packages that would be merged, in order:

Calculating dependencies -
!!! All ebuilds that could satisfy "~app-admin/apache-tools-2.2.8" have
been masked.
```

```
!!! One of the following masked packages is required to complete your re
quest:
- app-admin/apache-tools-2.2.8 (masked by: ~x86 keyword)

For more information, see MASKED PACKAGES section in the emerge man page
 or
refer to the Gentoo Handbook.
(dependency required by "www-servers/apache-2.2.8" [ebuild])
```

Sollte eine ähnliche Ausgabe erscheinen, befinden sich vermutlich noch die Spielereien aus Kapitel 5 in den Dateien `/etc/portage/package.use`, `/etc/portage/package.keywords` und `/etc/portage/package.unmask`. Wir wollen hier aber die stabile Version installieren und entfernen zunächst einmal wieder alle `www-servers/apache`-Einträge mit `nano` aus diesen Dateien.

Danach sollte die Ausgabe besser aussehen:

```
gentoo ~ # emerge -pu www-servers/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild     U ] dev-lang/perl-5.8.8-r4 [5.8.8-r2]
[ebuild  N    ] dev-util/pkgconfig-0.22  USE="-hardened"
[ebuild     U ] app-misc/mime-types-7 [5]
[ebuild     U ] sys-devel/autoconf-2.61-r1 [2.61]
[ebuild  NS   ] dev-libs/apr-1.2.11  USE="ipv6 -debug -doc -urandom"
[ebuild     U ] dev-libs/openssl-0.9.8g [0.9.8d] USE="-gmp% -kerberos%"
[ebuild  N    ] dev-libs/libpcre-7.4  USE="unicode -doc"
[ebuild     U ] sys-devel/libtool-1.5.24 [1.5.22] USE="-vanilla%"
[ebuild     U ] net-nds/openldap-2.3.39-r2 [2.3.30-r2]
[ebuild  NS   ] dev-libs/apr-util-1.2.10  USE="berkdb gdbm ldap mysql -d
oc -postgres -sqlite -sqlite3"
[ebuild     U ] www-servers/apache-2.2.6-r7 [2.0.58-r2] USE="-sni% -stat
ic% -suexec%" APACHE2_MODULES="actions%* alias%* auth_basic%* authn_alia
s%* authn_anon%* authn_dbm%* authn_default%* authn_file%* authz_dbm%* au
thz_default%* authz_groupfile%* authz_host%* authz_owner%* authz_user%*
autoindex%* cache%* dav%* dav_fs%* dav_lock%* deflate%* dir%* disk_cache
%* env%* expires%* ext_filter%* file_cache%* filter%* headers%* include%
* info%* log_config%* logio%* mem_cache%* mime%* mime_magic%* negotiatio
n%* rewrite%* setenvif%* speling%* status%* unique_id%* userdir%* usertr
ack%* vhost_alias%* -asis% -auth_digest% -authn_dbd% -cern_meta% -charse
t_lite% -dbd% -dumpio% -ident% -imagemap% -log_forensic% -proxy% -proxy_
ajp% -proxy_balancer% -proxy_connect% -proxy_ftp% -proxy_http% -version%
" APACHE2_MPMS="-event% -itk% -peruser% -prefork% -worker%"
[ebuild  N    ] app-admin/apache-tools-2.2.6  USE="ssl"
```

Es hat sich offensichtlich einiges seit dem `2007.0`-Release getan. Zum einen ist `>=www-servers/apache-2.2` mittlerweile die stabile Version, und bei

den USE-Flags hat es massive Veränderungen gegeben. Wir hatten schon in Kapitel 5.1.4 erwähnt, dass sich das Konzept der erweiterten USE-Flags (`USE_EXPAND`) weiter verbreitet, und der Apache-Server ist ein gutes Beispiel dafür.

Offensichtlich können wir jetzt bei der Installation bestimmen, welche Module der Apache-Server aktiviert haben soll. Das dafür zuständige, erweiterte USE-Flag ist `APACHE2_MODULES`. Auch die Multi-Processing-Module (siehe Seite 122) wurden aus den normalen USE-Flags herausgenommen und finden sich im erweiterten USE-Flag `APACHE2_MPMS` wieder.

Den Zeichen- bzw. Farbcode der USE-Flags haben wir auf Seite 114 genauer beschrieben.

Vielfach wollen wir nicht nur das Paket selbst, sondern auch gleich alle Abhängigkeiten des Paketes mit auf den neuesten Stand bringen. Die Option `--deep` (bzw. `-D`) instruiert `emerge`, auch für jedes im Baum der Abhängigkeiten vorliegende Paket zu überprüfen, ob neuere Versionen existieren:

```
gentoo ~ # emerge -puD www-servers/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild     U ] sys-devel/gnuconfig-20070724 [20060702]
[ebuild     U ] dev-libs/expat-2.0.1 [1.95.8]
[ebuild     U ] app-misc/pax-utils-0.1.16 [0.1.15]
[ebuild  N    ] app-admin/python-updater-0.2
[ebuild  N    ] dev-util/pkgconfig-0.22  USE="-hardened"
[ebuild     U ] app-misc/mime-types-7 [5]
[ebuild  N    ] dev-libs/libpcre-7.4  USE="unicode -doc"
[ebuild     U ] sys-apps/ed-0.8 [0.2-r6]
[ebuild     U ] sys-apps/man-pages-2.75 [2.42]
[ebuild     U ] sys-devel/binutils-config-1.9-r4 [1.9-r3]
[ebuild     U ] app-misc/ca-certificates-20070303-r1 [20061027.2]
[ebuild     U ] sys-process/procps-3.2.7 [3.2.6]
[ebuild     U ] dev-lang/perl-5.8.8-r4 [5.8.8-r2]
[ebuild  N    ] perl-core/Storable-2.16
[ebuild     U ] dev-perl/Net-Daemon-0.43 [0.39]
[ebuild     U ] virtual/perl-Storable-2.16 [2.15]
[ebuild     U ] dev-perl/PlRPC-0.2020-r1 [0.2018]
[ebuild     U ] dev-perl/DBI-1.601 [1.53]
[ebuild     U ] sys-devel/autoconf-2.61-r1 [2.61]
[ebuild     U ] sys-devel/libtool-1.5.24 [1.5.22] USE="-vanilla%"
[ebuild  NS   ] dev-libs/apr-1.2.11  USE="ipv6 -debug -doc -urandom"
[ebuild     U ] sys-libs/ncurses-5.6-r2 [5.5-r3] USE="-profile%"
[ebuild     U ] sys-libs/readline-5.2_p7 [5.1_p4]
[ebuild     U ] sys-libs/gpm-1.20.1-r6 [1.20.1-r5]
[ebuild     U ] dev-libs/openssl-0.9.8g [0.9.8d] USE="-gmp% -kerberos%"
[ebuild     U ] dev-db/mysql-5.0.54 [5.0.26-r2]
[ebuild     U ] dev-perl/DBD-mysql-4.00.5 [3.0008]
[ebuild  NS   ] sys-libs/db-4.5.20_p2  USE="-bootstrap -doc -java -nocxx
```

```
 -tcl -test"
[ebuild     U ] dev-lang/python-2.4.4-r6 [2.4.3-r4] USE="-examples% -not
hreads%"
[ebuild     U ] net-nds/openldap-2.3.39-r2 [2.3.30-r2]
[ebuild  NS   ] dev-libs/apr-util-1.2.10  USE="berkdb gdbm ldap mysql -d
oc -postgres -sqlite -sqlite3"
[ebuild  N    ] dev-libs/libxml2-2.6.30-r1  USE="ipv6 python readline -d
ebug -doc -test"
[ebuild     U ] sys-devel/gettext-0.17 [0.16.1] USE="acl%* openmp%*"
[ebuild     U ] sys-devel/m4-1.4.10 [1.4.7] USE="-examples%"
[ebuild     U ] sys-apps/diffutils-2.8.7-r2 [2.8.7-r1]
[ebuild     U ] sys-apps/man-1.6e-r3 [1.6d]
[ebuild     U ] sys-devel/binutils-2.18-r1 [2.16.1-r3]
[ebuild     U ] sys-apps/findutils-4.3.11 [4.3.2-r1]
[ebuild     U ] sys-apps/attr-2.4.39 [2.4.32]
[ebuild     U ] sys-apps/acl-2.2.45 [2.2.39-r1]
[ebuild     U ] net-misc/rsync-2.6.9-r5 [2.6.9-r1]
[ebuild     U ] sys-apps/coreutils-6.9-r1 [6.4] USE="-xattr%"
[ebuild     U ] www-servers/apache-2.2.6-r7 [2.0.58-r2] USE="-sni% -stat
ic% -suexec%" APACHE2_MODULES="actions%* alias%* auth_basic%* authn_alia
s%* authn_anon%* authn_dbm%* authn_default%* authn_file%* authz_dbm%* au
thz_default%* authz_groupfile%* authz_host%* authz_owner%* authz_user%*
autoindex%* cache%* dav%* dav_fs%* dav_lock%* deflate%* dir%* disk_cache
%* env%* expires%* ext_filter%* file_cache%* filter%* headers%* include%
* info%* log_config%* logio%* mem_cache%* mime%* mime_magic%* negotiatio
n%* rewrite%* setenvif%* speling%* status%* unique_id%* userdir%* usertr
ack%* vhost_alias%* -asis% -auth_digest% -authn_dbd% -cern_meta% -charse
t_lite% -dbd% -dumpio% -ident% -imagemap% -log_forensic% -proxy% -proxy_
ajp% -proxy_balancer% -proxy_connect% -proxy_ftp% -proxy_http% -version%
" APACHE2_MPMS="-event% -itk% -peruser% -prefork% -worker%"
[ebuild  N    ] app-admin/apache-tools-2.2.6  USE="ssl"
```

Im Unterschied zur Liste weiter oben sind eine ganze Menge neuer Pakete hinzugekommen. Verzichten wir auf die `--deep`-Option, so aktualisiert `emerge` nur die Pakete, die wirklich zwingend einer Aktualisierung bedürfen, damit die von uns gewünschte Software einwandfrei läuft.

Wer es ganz genau nimmt kann auch noch die Paketabhängigkeiten aktualisieren, die nicht unbedingt zur Laufzeit des Programms, sondern nur zum Kompilieren bzw. in der Installationsphase zwingend notwendig sind (*Build time dependencies* im Gegensatz zu den normalen *Run time dependencies*; siehe Seite 323). Dazu dient die Option `--with-bdeps y`. Sie kommt eher selten zum Einsatz.

In den obigen Beispielen sind zu aktualisierende Pakete mit der Markierung `[ebuild U ]` versehen, wobei hier das `U` „update“ bezeichnet. Diese Pakete liegen also im synchronisierten Portage-Baum in einer aktualisierten Version vor. Ersichtlich wird das auch an den Versionsnummern, die hinter den Namen der Ebuilds aufgeführt werden. So ist z. B. dem Ausschnitt zu dem Paket `dev-libs/openssl` (`dev-libs/openssl-0.9.8g [0.9.8d]`) zu entnehmen, dass von der derzeitigen Version `0.9.8d` (die Angabe hinter dem

Ebuild-Namen, in blau und von eckigen Klammern eingerahmt) auf die Version `0.9.8g` (die Angabe direkt hinter dem Paketnamen) aktualisiert werden soll.

Wer wissen möchte, was die Gentoo-Entwickler bei einem Paket in der Zwischenzeit geändert haben, bevor er es aktualisiert, kann sich über die Option `--changelog` (bzw. `-l`) die Datei `ChangeLog` des entsprechenden Ebuilds anzeigen lassen. Wir nehmen hier der Einfachheit halber ein wenig verändertes Paket:

```
gentoo ~ # emerge -puvl app-misc/pax-utils

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild     U ] app-misc/pax-utils-0.1.16 [0.1.15] USE="-caps" 64 kB

Total: 1 package (1 upgrade), Size of downloads: 64 kB

*pax-utils-0.1.16

  24 Aug 2007; <solar@gentoo.org> -pax-utils-0.1.13.ebuild,
  -pax-utils-0.1.14.ebuild, +pax-utils-0.1.16.ebuild:
  - Version bump. man pages moved over to docbook. New: endian and perm
  displays.. New: when -Tv are used together the disasm will be displaye
d of
  the offending text rel. The pax-utils code should compile out of the b
ox on
  solaris now. Lots of misc fixes.. to many to list..

  01 Mar 2007; <genstef@gentoo.org> pax-utils-0.1.13.ebuild,
  pax-utils-0.1.14.ebuild, pax-utils-0.1.15.ebuild:
  Dropped ppc-macos keyword, see you in prefix

  03 Feb 2007; Bryan Østergaard <kloeri@gentoo.org>
  pax-utils-0.1.15.ebuild:
  Stable on Alpha, bug 163453.

  02 Feb 2007; Alexander H. Færøy <eroyf@gentoo.org>
  pax-utils-0.1.15.ebuild:
  Stable on MIPS; bug #163453

  31 Jan 2007; Markus Rothe <corsair@gentoo.org> pax-utils-0.1.15.ebuild:
  Stable on ppc64; bug #163453

  30 Jan 2007; Steve Dibb <beandog@gentoo.org> pax-utils-0.1.15.ebuild:
  amd64 stable, bug 163453

  25 Jan 2007; Gustavo Zacarias <gustavoz@gentoo.org>
  pax-utils-0.1.15.ebuild:
  Stable on sparc wrt #163453
```

```
  24 Jan 2007; Jeroen Roovers <jer@gentoo.org> pax-utils-0.1.15.ebuild:
  Stable for HPPA (bug #163453).

  23 Jan 2007; Raúl Porcel <armin76@gentoo.org> pax-utils-0.1.15.ebuild:
  x86 stable wrt bug 163453

  23 Jan 2007; nixnut <nixnut@gentoo.org> pax-utils-0.1.15.ebuild:
  Stable on ppc wrt bug 163453
```

Aktualisieren wir an dieser Stelle einmal den Apache-Server mit `emerge -u www-servers/apache`, damit wir in diesem Gebiet auf dem neuesten Stand sind und etwas herumexperimentieren können.

Verwendet man `emerge` mit der `--update`-Option, so werden Veränderungen bei den USE-Flags nicht mit einbezogen. So führt folgender Aufruf zu keiner Aktion, obwohl wir hier mit `USE="-ssl"` explizit angeben, dass wir das `www-servers/apache`-Paket diesmal ohne SSL-Unterstützung installieren möchten.

```
gentoo ~ # USE="-ldap" emerge -puv www-servers/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!

Total size of downloads: 0 kB
```

Um bei einem Update auch auf Veränderungen der USE-Flags zu reagieren, fügen wir die Option `--newuse` (bzw. `-N`) hinzu:

```
gentoo ~ # USE="-ldap" emerge -puvN www-servers/apache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] sys-libs/zlib-1.2.3-r1  USE="(-build%)" 0 kB
[ebuild   R   ] dev-libs/apr-util-1.2.10  USE="berkdb gdbm mysql -doc -l
dap* -postgres -sqlite -sqlite3" 0 kB
[ebuild   R   ] www-servers/apache-2.2.6-r7  USE="ssl -debug -doc -ldap*
 (-selinux) -sni -static -suexec -threads" APACHE2_MODULES="actions alia
s auth_basic authn_alias authn_anon authn_dbm authn_default authn_file a
uthz_dbm authz_default authz_groupfile authz_host authz_owner authz_user
 autoindex cache dav dav_fs dav_lock deflate dir disk_cache env expires
ext_filter file_cache filter headers include info log_config logio mem_c
ache mime mime_magic negotiation rewrite setenvif speling status unique_
id userdir usertrack vhost_alias -asis -auth_digest -authn_dbd -cern_met
a -charset_lite -dbd -dumpio -ident -imagemap -log_forensic -proxy -prox
y_ajp -proxy_balancer -proxy_connect -proxy_ftp -proxy_http -version" AP
ACHE2_MPMS="-event -itk -peruser -prefork -worker" 0 kB

Total: 3 packages (3 reinstalls), Size of downloads: 0 kB
```

Der Ebuild ist jetzt mit der Markierung `[ebuild R ]` („re-emerge“) versehen. Außerdem kennzeichnet `emerge` das veränderte USE-Flag jetzt mit einem Sternchen.

Damit kennen wir alle notwendigen Befehle, um unser System zu aktualisieren. Es ist allerdings nicht sinnvoll, das Update für jedes Paket einzeln durchzugehen. Es wäre deutlich angenehmer, alle auf dem System installierten Pakete in einem Arbeitsschritt zu aktualisieren. Diese Funktion erfüllt der spezielle Paketname `world`:

```
gentoo ~ # emerge -pvuN world

These are the packages that would be merged, in order:

Calculating world dependencies... done!
[ebuild   R   ] sys-libs/zlib-1.2.3-r1  USE="(-build%)" 0 kB
[ebuild     U ] dev-libs/expat-2.0.1 [1.95.8] USE="(-test%)" 0 kB
[ebuild     U ] sys-devel/gcc-config-1.4.0-r4 [1.3.14] 0 kB
[ebuild     U ] dev-libs/gmp-4.2.2 [4.2.1] USE="-doc -nocxx" 1,707 kB
[ebuild  N    ] app-admin/python-updater-0.2  0 kB
[ebuild  N    ] dev-util/unifdef-1.20  65 kB
[ebuild     U ] app-arch/cpio-2.9-r1 [2.6-r5] USE="nls" 741 kB
[ebuild     U ] app-misc/pax-utils-0.1.16 [0.1.15] USE="-caps" 0 kB
[ebuild     U ] sys-libs/timezone-data-2007j [2007c] USE="nls" 345 kB
[ebuild  N    ] dev-libs/eventlog-0.2.5  305 kB
[ebuild     U ] app-arch/bzip2-1.0.4-r1 [1.0.3-r6] USE="-static (-build
%)" 822 kB
[ebuild     U ] sys-apps/hdparm-7.7 [6.6] 62 kB
[ebuild     U ] sys-apps/ethtool-6 [4] 127 kB
[ebuild     U ] net-misc/dhcpcd-3.1.5-r1 [2.0.5-r1] USE="-vram% (-build%
) (-debug%) (-static%)" 40 kB
[ebuild     U ] sys-kernel/linux-headers-2.6.23-r3 [2.6.17-r2] USE="(-gc
c64%)" 4,671 kB
[ebuild     U ] sys-apps/debianutils-2.28.2 [2.17.4] USE="-static" 135 k
B
[ebuild     U ] dev-libs/mpfr-2.3.0_p3 [2.2.0_p16] 853 kB
[ebuild     U ] sys-apps/ed-0.8 [0.2-r6] 67 kB
[ebuild     U ] sys-apps/pciutils-2.2.8 [2.2.3-r2] USE="zlib%* -network-
cron%" 228 kB
[ebuild     U ] app-portage/flagedit-0.0.7 [0.0.5] 6 kB
[ebuild     U ] sys-apps/sysvinit-2.86-r10 [2.86-r8] USE="(-ibm) (-selin
ux) -static" 0 kB
[ebuild     U ] net-misc/iputils-20070202 [20060512] USE="ipv6 -doc -sta
tic" 87 kB
[ebuild     U ] sys-apps/man-pages-2.75 [2.42] USE="nls" 1,815 kB
[ebuild     U ] sys-devel/binutils-config-1.9-r4 [1.9-r3] 0 kB
[ebuild     U ] sys-process/procps-3.2.7 [3.2.6] USE="(-n32)" 276 kB
[ebuild     U ] sys-fs/udev-115-r1 [104-r12] USE="(-selinux)" 210 kB
[ebuild     U ] sys-libs/ncurses-5.6-r2 [5.5-r3] USE="gpm unicode -boots
trap -build -debug -doc -minimal -nocxx -profile% -trace" 2,353 kB
[ebuild     U ] sys-libs/readline-5.2_p7 [5.1_p4] 2,008 kB
[ebuild     U ] sys-libs/gpm-1.20.1-r6 [1.20.1-r5] USE="(-selinux)" 9 kB
```

```
[ebuild     U ] sys-apps/less-416 [394] USE="unicode" 288 kB
[ebuild     U ] dev-db/mysql-5.0.54 [5.0.26-r2] USE="berkdb perl ssl -bi
g-tables -cluster -debug -embedded -extraengine -latin1 -max-idx-128 -mi
nimal (-selinux) -static" 26,860 kB
[ebuild     U ] dev-perl/DBD-mysql-4.00.5 [3.0008] 120 kB
[ebuild     U ] dev-lang/python-2.4.4-r6 [2.4.3-r4] USE="berkdb gdbm ipv
6 ncurses readline ssl -bootstrap -build -doc -examples% -nocxx -nothre
ads% -tk -ucs2" 7,977 kB
[ebuild  N    ] dev-libs/libxml2-2.6.30-r1  USE="ipv6 python readline -d
ebug -doc -test" 4,616 kB
[ebuild     U ] sys-libs/cracklib-2.8.10 [2.8.9-r1] USE="nls python" 565
 kB
[ebuild     U ] sys-apps/file-4.21-r1 [4.20-r1] USE="python" 538 kB
[ebuild     U ] sys-devel/gettext-0.17 [0.16.1] USE="acl%* nls openmp%*
 -doc -emacs -nocxx" 11,369 kB
[ebuild     U ] sys-devel/binutils-2.18-r1 [2.16.1-r3] USE="nls -multisl
ot -multitarget -test -vanilla" 14,629 kB
[ebuild     U ] sys-devel/flex-2.5.33-r3 [2.5.33-r1] USE="nls -static" 0
 kB
[ebuild     U ] sys-devel/m4-1.4.10 [1.4.7] USE="nls -examples%" 722 kB
[ebuild     U ] sys-apps/acl-2.2.45 [2.2.39-r1] USE="nls (-nfs)" 150 kB
[ebuild     U ] sys-apps/man-1.6e-r3 [1.6d] USE="nls" 247 kB
[ebuild     U ] sys-apps/findutils-4.3.11 [4.3.2-r1] USE="nls (-selinux)
 -static" 2,003 kB
[ebuild     U ] sys-apps/grep-2.5.1a-r1 [2.5.1-r8] USE="nls pcre%* -stat
ic (-build%)" 516 kB
[ebuild     U ] sys-apps/gawk-3.1.5-r5 [3.1.5-r2] USE="nls" 0 kB
[ebuild     U ] app-editors/nano-2.0.7 [2.0.2] USE="ncurses nls unicode
-debug -justify -minimal -slang -spell" 1,332 kB
[ebuild     U ] dev-util/dialog-1.1.20071028 [1.0.20050206] USE="nls%* u
nicode -examples%" 362 kB
[ebuild     U ] app-arch/gzip-1.3.12 [1.3.5-r10] USE="nls -pic -static (
-build%)" 452 kB
[ebuild     U ] sys-apps/net-tools-1.60-r13 [1.60-r12] USE="nls -static"
 105 kB
[ebuild     U ] sys-apps/kbd-1.13-r1 [1.12-r8] USE="nls" 652 kB
[ebuild     U ] app-arch/tar-1.19-r1 [1.16-r2] USE="nls -static" 1,839 k
B
...
```

`world` umfasst alle System-Pakete und jene, die wir manuell hinzugefügt haben. Letztere verwaltet Portage in der Datei `/var/lib/portage/world`:

```
gentoo ~ # cat /var/lib/portage/world
app-admin/eselect
app-admin/showconsole
app-admin/syslog-ng
app-benchmarks/bootchart
app-misc/livecd-tools
app-portage/emerge-delta-webrsync
app-portage/euses
```

```
app-portage/flagedit
app-portage/gentoolkit
app-portage/mirrorselect
dev-db/mysql
net-dialup/ppp
net-misc/dhcpcd
sys-apps/ethtool
sys-apps/netplug
sys-boot/grub
sys-kernel/genkernel
sys-kernel/gentoo-sources
sys-process/vixie-cron
www-servers/apache
```

Die Liste enthält nur die Pakete, die wir tatsächlich auf der Kommandozeile in Verbindung mit `emerge` angegeben haben, nicht jedoch solche, die `emerge` aufgrund von Abhängigkeiten installiert hat.

Um ein Paket zum Testen zu installieren, ohne es gleich in diese Liste fest installierter Pakete aufzunehmen, kann man `emerge` mit der Option `--oneshot` (bzw. `-1`) aufrufen. Damit installieren wir das Paket zwar normal, fügen es aber eben nicht zu `world` hinzu.

Wollen wir ein so installiertes Paket bzw. ein als Abhängigkeit installiertes Paket, das noch in der `world`-Datei fehlt, hinzufügen, ohne das bereits installierte Paket erneut zu kompilieren und installieren, erreichen wir das über die Option `--noreplace` (bzw. `-n`) Diese dient dazu, Pakete wirklich nur dann zu installieren, wenn sie noch nicht installiert wurden. Allerdings nimmt `emerge` das Paket auf jeden Fall in `world` auf, wenn es dort noch fehlt:

```
gentoo ~ # emerge -n net-nds/openldap
Calculating dependencies... done!
>>> Recording net-nds/openldap in ''world'' favorites file...
>>> Auto-cleaning packages...

>>> No outdated packages were found on your system.
gentoo ~ # grep ldap /var/lib/portage/world
net-nds/openldap
```

`emerge world` ist in jedem Fall der Befehl, den wir für ein volles Update unseres Systems verwenden. Die Option `--update` sollte unbedingt gesetzt sein, damit `emerge` wirklich nur Pakete aktualisiert, die ein Update nötig haben. Die Option `--newuse` ist ebenfalls notwendig, damit Portage alle Pakete mit veränderten USE-Flags aktualisiert. Abschließend brauchen wir noch die Option `--deep`, damit `emerge` alle Abhängigkeiten ebenfalls berücksichtigt und nicht nur die Pakete, die direkt in unserer `world`-Datei gelistet sind. Damit ergibt sich folgender Befehl für ein komplettes Update des Systems:

```
emerge --update --newuse --deep world
```

Alternativ kann man natürlich auch die Kurzform `emerge -uND world` verwenden.

Abgesehen von `world` gibt es noch den speziellen Bezeichner `system`, der alle Pakete umfasst, die für das Basissystem notwendig sind. Diese Liste ist über das installierte Profil definiert. Allerdings hat der Befehl `emerge system` selten praktische Bedeutung.

10.3 Konfiguration aktualisieren

Der Update-Vorgang endet in den meisten Fällen mit einigen allgemeinen Informationen und häufig mit einer Nachricht über Konfigurationsdateien, die ein Update benötigen:

```
 * Regenerating GNU info directory index...
 * Processed 96 info files.
 * IMPORTANT: 5 config files in /etc need updating.
 * Type emerge --help config to learn how to update config files.
```

Dies ist nicht zwingend bei jedem Update der Fall, aber sehr häufig bringen neue Software-Versionen natürlich auch neue Konfigurationsoptionen mit sich.

Gentoo bietet hier mit der *Config Protection* ein besonderes Feature, das den Benutzer davor bewahrt, bei der Aktualisierung von Paketen unbedacht wichtige Konfigurationsdateien zu überschreiben. Da die Konfiguration des Systems ein zentrales Element für den einwandfreien Betrieb der Maschine darstellt, wollen wir uns etwas ausführlicher mit der Aktualisierung der Konfiguration beschäftigen.

Config Protection arbeitet verzeichnisorientert: Es lassen sich also bestimmte Verzeichnisse definieren, in denen Portage bei der Installation eines Paketes bereits vorhandene Dateien nicht überschreibt. Standardmäßig gilt dies z. B. für das Verzeichnis `/etc`. Die geschützten Verzeichnisse lassen sich über die Variable `CONFIG_PROTECT` in der Datei `/etc/make.conf` festlegen. Die Basisdefinition `CONFIG_PROTECT="/etc"` findet sich in der Datei `/etc/make.globals`. Alle Verzeichnisse, die wir der gleichen Variable in `/etc/make.conf` hinzufügen, addieren sich zu dieser Basis-Definition.

Die Möglichkeit, den Schutz für einzelne Verzeichnisse wieder zu entfernen, bietet dann im Gegenzug die Variable `CONFIG_PROTECT_MASK`, die sich ebenfalls in `/etc/make.conf` definieren lässt. Als Standard ist (ebenfalls in `/etc/make.globals`) das Verzeichnis `/etc/env.d` vom Schutz ausgeschlossen. In diesem Verzeichnis können Ebuilds, wie unter 1.5.1 schon

beschrieben, paketspezifische Umgebungsvariablen festlegen. Da jeder Benutzer diese Umgebungsvariablen in seinen eigenen Bash-Startup-Skripten umdefinieren kann (und soll), würde ein Schutz der Dateien in `/etc/env.d` keinen Sinn ergeben, da sie nur die Standard-Konfiguration darstellen.

Es ist z. B. sinnvoll, das Verzeichnis `/etc/init.d` in die Variable `CONFIG_PROTECT_MASK` aufzunehmen, da die Init-Skripte, die in diesem Verzeichnis liegen, keine Konfigurationsdateien im eigentlichen Sinn darstellen. Die meisten Nutzer werden wohl auch keine eigenen Veränderungen an diesen Skripten vornehmen. Wichtig ist allerdings, als Nutzer genau zu wissen, für welche Verzeichnisse die *Config Protection* gilt und für welche nicht. Entfernt man den Schutz, überschreibt Portage gnadenlos alte Dateien, ohne darauf Rücksicht zu nehmen, ob der Benutzer an diesen Veränderungen vorgenommen hat oder nicht.

Wir gehen an dieser Stelle zurück zu unserem 2007.0-System und führen die folgenden Beispiele auf Basis des nicht aktualisierten Systems durch.

Schauen wir uns als konkretes Beispiel für die Aktualisierung von Konfigurationsdateien einmal das Paket `dev-libs/openssl` an. Da wir hier vom nicht aktualisierten System ausgehen, fehlt uns ein zu aktualisierendes Paket, aber das können wir umgehen, indem wir `dev-lib/openssl` *downgraden*, die derzeit installierte Version also reduzieren. Vorher schauen wir uns noch den Inhalt des zum Paket `dev-libs/openssl` gehörenden Konfigurationsverzeichnisses `/etc/ssl` an:

```
gentoo ~ # ls -la /etc/ssl/
insgesamt 44
drwxr-xr-x  5 root root  4096 29. Jan 20:25 .
drwxr-xr-x 41 root root  4096 29. Jan 20:25 ..
drwxr-xr-x  2 root root 12288 29. Jan 20:25 certs
drwxr-xr-x  2 root root  4096 29. Jan 20:25 misc
-rw-r--r--  1 root root  9374 29. Jan 20:25 openssl.cnf
drwx------  2 root root  4096  6. Apr 2007  private
gentoo ~ # emerge -av =dev-libs/openssl-0.9.7l

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild     UD] dev-libs/openssl-0.9.7l [0.9.8d] USE="zlib -bindist -ema
cs -test (-sse2%)" 2,645 kB

Total: 1 package (1 downgrade), Size of downloads: 2,645 kB

Would you like to merge these packages? [Yes/No] Yes
...

>>> No outdated packages were found on your system.
 * GNU info directory index is up-to-date.
 * IMPORTANT: 3 config files in '/etc' need updating.
 * Type emerge --help config to learn how to update config files.
```

Auch wenn wir unsere Version hier verringern: `emerge` beendet mit dem Hinweis, dass wir drei Konfigurationsdateien aktualisieren müssen. Offensichtlich gibt es zwischen `openssl-0.9.7` und `openssl-0.9.8` Unterschiede in der Standardkonfiguration.

Im Beispiel hat `emerge` Konfigurationsdateien im Verzeichnis `/etc/ssl` aktualisiert. Portage darf in `/etc` aber keine Dateien überschreiben und entscheidet sich dafür, die neu zu installierenden Dateien erst einmal umzubenennen und es dem Nutzer zu überlassen, das Update der Konfigurationsdateien manuell durchzuführen. Dies geschieht natürlich nur, wenn wirklich Unterschiede zwischen den alten und neuen Dateien bestehen. Sind neue und alte Version einer Datei identisch, ignoriert `emerge` die neue Datei automatisch.

So findet sich nach der Installation des älteren `dev-libs/openssl`-Pakets Folgendes im Verzeichnis `/etc/ssl` wieder:

```
gentoo ~ # ls -la /etc/ssl/
insgesamt 52
drwxr-xr-x  5 root root  4096 29. Jan 20:25 .
drwxr-xr-x 41 root root  4096 29. Jan 20:25 ..
drwxr-xr-x  2 root root 12288 29. Jan 20:25 certs
-rw-r--r--  1 root root  9381 29. Jan 20:25 ._cfg0000_openssl.cnf
drwxr-xr-x  2 root root  4096 29. Jan 20:25 misc
-rw-r--r--  1 root root  9374 29. Jan 20:25 openssl.cnf
drwx------  2 root root  4096  6. Apr 2007  private
```

Im Vergleich zum weiter oben abgebildeten Datei-Listing sehen wir eine Datei mit Namen `._cfg0000_openssl.cnf`. Diese enthält die aktualisierte Version der Datei `openssl.cnf`. Das Präfix der Datei `._cfg0000_` deutet darauf hin, dass wir es hier mit einer noch nicht aktualisierten Konfigurationsdatei zu tun haben. Die Tatsache, dass die Datei mit einem Punkt (`.`) im Dateinamen beginnt, bedeutet, dass die Datei versteckt ist und das System sie im Normalfall nicht anzeigt.

Solange solche Dateien in den geschützten Verzeichnissen vorhanden sind, zeigt Portage am Ende eines `emerge`-Laufes den Hinweis `X config files in /etc need updating` an. Es liegt dann beim Nutzer, die alte und die neue Version zusammenzuführen. Sinn dieses Vorgangs ist es natürlich, Veränderungen, die der Nutzer in der vorigen Version vorgenommen hat, an dieser Stelle sicher in die Konfiguration der neuen Version zu überführen.

Derzeit bietet Gentoo drei verschiedene Tools für das Update der Konfigurationsdateien: `etc-update`, `dispatch-conf` und `cfg-update`. Sowohl `etc-update` als auch `dispatch-conf` sind integraler Bestandteil des `sys-apps/portage`-Paketes, während `cfg-update` als eigenes Paket gleichen Namens (`app-portage/cfg-update`) installiert werden muss.

`etc-update` ist das älteste Tool, `dispatch-conf` liefert als neuere Variante eine vereinfachte Handhabung und mehr Funktionalität. `cfg-update` ist in den Funktionen vergleichbar mit `dispatch-conf`, bietet jedoch besseren Support für grafische `diff`-Programme auf Desktop-Maschinen. Von daher bringt es auf unserer Maschine keinen Vorteil, und es ist zudem noch als instabil markiert. Wir verwenden es hier nicht und konzentrieren uns nur auf `dispatch-conf`.

Die Einstellungen für das Tool `dispatch-conf` finden sich in der Datei `/etc/dispatch-conf.conf`. Die Standardeinstellungen dürften für die meisten Nutzer in Ordnung sein, aber wer möchte kann dem Werkzeug weitere Möglichkeiten zum automatischen Kombinieren von alter und neuer Konfigurationsdatei geben. Per Default übernimmt `dispatch-conf` Veränderungen, die Leerzeilen oder Kommentarzeilen betreffen, nicht automatisch, sondern fragt den Nutzer, ob er dies möchte. Das lässt sich in der Konfigurationsdatei über die Einstellung `replace-wscomments` modifizieren:

```
replace-wscomments=yes
```

Darüber hinaus kann `dispatch-conf` auch automatisch Dateien zusammenführen, wenn der User die Datei selbst noch nie modifiziert hat:

```
replace-unmodified=yes
```

Das Tool `dispatch-conf` kann automatisch Backup-Dateien im angegebenen `archive-dir` anlegen. Das ist sinnvoll, damit die automatischen Aktionen keinen Schaden anrichten und jederzeit rückgängig zu machen sind.

Die Standardeinstellung `archive-dir=/etc/config-archive` muss man nicht verändern. Allerdings legt `dispatch-conf` dort erst dann Backup-Dateien an, wenn das Verzeichnis auch existiert. Bei der Installation älterer Versionen des Paketes `sys-apps/portage` wird es nicht automatisch angelegt.

```
gentoo ~ # mkdir /etc/config-archive
```

Wer eine farbige Ausgabe des Tools `dispatch-conf` bevorzugt, kann den `diff`-Befehl in `/etc/dispatch-conf.conf` auch in `colordiff` umwandeln:

```
diff="colordiff -Nu '%s' '%s' | less --no-init --QUIT-AT-EOF"
```

Dazu müssen wir aber noch das entsprechende Werkzeug installieren:

```
gentoo ~ # emerge -av app-misc/colordiff

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-misc/colordiff-1.0.6-r1  16 kB

Total: 1 package (1 new), Size of downloads: 16 kB

Would you like to merge these packages? [Yes/No] Yes
```

`dispatch-conf` selbst ruft man im Normalfall ohne weitere Optionen auf. Das Programm durchsucht dann die geschützten Verzeichnisse nach Konfigurations-Updates. Es ist möglich, die Aktion von `dispatch-conf` auf ein bestimmtes Verzeichnis zu beschränken, indem man dieses als Argument übergibt:

```
gentoo ~ # dispatch-conf /etc/ssl
```

Das Tool zeigt dann im typischen `diff`-Stil die Unterschiede zwischen neuer und alter Konfigurationsdatei an:

```
--- /etc/ssl/openssl.cnf        2008-01-30 08:32:45.000000000 +0100
+++ /etc/ssl/._cfg0000_openssl.cnf      2008-01-29 20:25:43.000000000 +0
100
@@ -44,8 +44,8 @@

 certificate    = $dir/cacert.pem       # The CA certificate
 serial         = $dir/serial           # The current serial number
-crlnumber      = $dir/crlnumber        # the current crl number
-                                       # must be commented out to leave
 a V1 CRL
+#crlnumber     = $dir/crlnumber        # the current crl number must be
+                                       # commented out to leave a V1 CR
L
 crl            = $dir/crl.pem          # The current CRL
 private_key    = $dir/private/cakey.pem# The private key
 RANDFILE       = $dir/private/.rand    # private random number file
@@ -67,7 +67,7 @@

 default_days   = 365                   # how long to certify for
 default_crl_days= 30                   # how long before next CRL
-default_md     = sha1                  # which md to use.
+default_md     = md5                   # which md to use.
 preserve       = no                    # keep passed DN ordering

 # A few difference way of specifying how similar the request should loo
k
@@ -188,7 +188,7 @@
```

```
 # PKIX recommendations harmless if included in all certificates.
 subjectKeyIdentifier=hash
-authorityKeyIdentifier=keyid,issuer
+authorityKeyIdentifier=keyid,issuer:always

 # This stuff is for subjectAltName and issuerAltname.
 # Import the email address.

>> (1 of 3) -- /etc/ssl/openssl.cnf
>> q quit, h help, n next, e edit-new, z zap-new, u use-new
   m merge, t toggle-merge, l look-merge:
```

Mit der Taste `[U]` für `use-new` würden wir die neue Konfigurationsdatei akzeptieren und verwenden. Wir haben an dieser Stelle aber eine alte Version installiert, was wir in Kürze wieder rückgängig machen werden, und sollten hier erst einmal alle Aktualisierungen mit `[Z]` ablehnen.

Die Funktionsweise des Tools lässt sich besser an einem kurzen, konstruierten Beispiel verstehen. Um das Folgende nachvollziehen, ist die bereits besprochene Einstellung in `/etc/dispatch-conf.conf` vorzunehmen:

```
replace-wscomments=yes
replace-unmodified=yes
archive-dir=/etc/config-archive
```

Wir erstellen jetzt ein temporäres Verzeichnis unter `/tmp` und erzeugen eine hypothetische Konfigurationsdatei:

```
gentoo ~ # mkdir /tmp/d-c
gentoo ~ # echo "
> # OPTION1
> option1=test1
> # OPTION2
> option2=test2
> " > /tmp/d-c/config
```

Angenommen wir würden das hypothetische Paket nun aktualisieren, wobei die Konfiguration jedoch nicht verändert wurde. Wir simulieren das, indem wir die gerade erstellte Demo-Konfiguration in eine versteckte Konfigurationsdatei kopieren. Die Datei benennen wir so, wie Portage es nach der Installation eines Paketes tun würde: `._cfg0001_` als Präfix, gefolgt von dem Dateinamen der zugehörigen Konfigurationsdatei:

```
gentoo ~ # cp /tmp/d-c/config /tmp/d-c/._cfg0001_config
```

Unter der Annahme, dass wir es `dispatch-conf` mit `replace-unmodifed=yes` erlaubt haben, unveränderte Konfigurationsdateien automatisch zu aktualisieren, erfordert der folgende Aufruf keine User-Interaktion:

```
gentoo ~ # dispatch-conf /tmp/d-c
```

Die von uns angelegte, versteckte Datei verschwindet, und dafür speichert `dispatch-conf` beim ersten Aufruf die Originalkonfiguration im `archive-dir`, das wir in `/etc/dispatch-conf.conf` festgelegt haben:

```
gentoo ~ # ls -la /tmp/d-c/
total 14
drwxr-xr-x 2 root root   72 Nov 30 13:50 .
drwxrwxrwt 8 root root 9760 Nov 30 13:50 ..
-rw-r--r-- 1 root root   51 Nov 30 13:45 config
gentoo ~ # ls -la /etc/config-archive/tmp/d-c/
total 8
drwxr-xr-x 2 root root 104 Nov 30 13:50 .
drwxr-xr-x 3 root root  72 Nov 30 13:50 ..
-rw-r--r-- 1 root root  51 Nov 30 13:45 config
-rw-r--r-- 1 root root  51 Nov 30 13:45 config.dist
```

Testen wir die gleiche Aktion, aber diesmal simulieren wir einige hinzugefügte Kommentarzeilen in der neuen Version der Konfigurationsdatei:

```
gentoo ~ # cp /tmp/d-c/config /tmp/d-c/._cfg0001_config
gentoo ~ # echo "# OPTION3" >> /tmp/d-c/._cfg0001_config
gentoo ~ # echo "#option3=test3" >> /tmp/d-c/._cfg0001_config
gentoo ~ # dispatch-conf /tmp/d-c
```

Wenn wir es `dispatch-conf` mit `replace-wscomments=yes` erlaubt haben, Kommentare, Leerzeilen und unmodifizierte Sektionen automatisch zu aktualisieren, bleibt auch diesmal, wie zu erwarten, eine Reaktion aus. `dispatch-conf` entfernt die Datei `._cfg0001_config` und fügt der eigentlichen Konfigurationsdatei `config` die hinzugefügten Kommentarzeilen hinzu:

```
gentoo ~ # cat /tmp/d-c/config

# OPTION1
option1=test1
# OPTION2
option2=test2

# OPTION3
#option3=test3
```

Bisher haben wir als Benutzer noch keinerlei Veränderung an der Konfigurationsdatei vorgenommen. Ohne Veränderungen an den Konfigurationsdateien sind die Aktionen von `dispatch-conf` allerdings nicht sonderlich spannend.

Kommen wir also zu einem realistischeren Beispiel. Wir verändern nun als Benutzer die Variablen `option1` und `option3` in unserer Konfigurationsdatei:

```
gentoo ~ # echo "
> # OPTION1
> option1=modifiziert1
> # OPTION2
> option2=test2
>
> # OPTION3
> option3=modifiziert3" > /tmp/d-c/config
```

Zunächst noch einmal eine unspektakuläre Variante: Angenommen bei der Aktualisierung des Paketes auf die nächste Version kam es zu keinen Veränderungen an der Konfigurationsdatei – d. h. die neu installierte Datei `._cfg0001_config` entspricht der Konfigurationsdatei der früheren Version, die `dispatch-conf` im `config-archive` unter `config.dist` abgespeichert hat –, dann wird `dispatch-conf` registrieren, dass es wieder nicht handeln muss. Wir kopieren also die gespeicherte Konfiguration aus dem Archiv von `dispatch-conf`, um die neue, aber unveränderte Konfigurationsdatei zu simulieren:

```
gentoo ~ # cp /etc/config-archive/tmp/d-c/config.dist \
> /tmp/d-c/._cfg0001_config
gentoo ~ # dispatch-conf /tmp/d-c
```

Wieder verrichtet `dispatch-conf` den Dienst klaglos, da sich die Konfigurationsdateien zwischen beiden Versionen offensichtlich nicht geändert haben und somit die Veränderungen des Nutzers unverändert bestehen bleiben können. Im Grunde löscht `dispatch-conf` hier einfach die Datei `._cfg0001_config` als irrelevant.

Beim nächsten Versionssprung verändert sich aber nun ein Wert, der im Konflikt mit den Änderungen des Benutzers steht. `dispatch-conf` kann also nicht mehr alleine entscheiden, welche die korrekte Konfiguration sein könnte, und fordert an dieser Stelle den User zur Interaktion auf. Dazu konstruieren wir eine Veränderung im nächsten simulierten Update: `option2` setzen wir jetzt auf einen neuen Wert.

```
gentoo ~ # echo "
> # OPTION1
> option1=test1
> # OPTION2
> option2=neu2
>
> # OPTION3
> #option3=test3" > /tmp/d-c/._cfg0001_config
```

Jetzt wird die Ausgabe von dispatch-conf endlich etwas interessanter, da es das erste Mal wirkliche Veränderungen zu kombinieren gilt:

```
gentoo ~ # dispatch-conf /tmp/d-c
--- /tmp/d-c/config     2006-11-30 16:01:39.000000000 +0100
+++ /tmp/d-c/._mrg0001_config   2006-11-30 16:01:46.000000000 +0100
@@ -2,7 +2,7 @@
 # OPTION1
 option1=modifiziert1
 # OPTION2
-option2=test2
+option2=neu2

 # OPTION3
 option3=modifiziert3

>> (1 of 1) -- /tmp/d-c/config
>> q quit, h help, n next, e edit-new, z zap-new, u use-new
   m merge, t toggle-merge, l look-merge:
```

dispatch-conf reduziert den Aufwand auf die einzige Variable, bei der es beim fingierten Update zu einer Änderung kam. Da option1 und option3 gleich geblieben sind, übernimmt dispatch-conf die Änderungen des Nutzers in die neue Datei. etc-update würde an allen drei Stellen eine Entscheidung des Nutzers verlangen.

An dieser Stelle können wir nun wählen, ob wir alle Veränderungen mit [Z] ablehnen möchten oder sie mit [U] akzeptieren. Mit der Taste [N] verschiebt man das Update auf später. Die merge-Operationen ([M]) sind für komplexere Kombinationen gedacht und erlauben bei jeder Veränderung zu entscheiden, ob man die alte oder die neue Version bevorzugt. In diesen Fällen hat man auch die Möglichkeit einen Editor aufzurufen, mit dem man jede Veränderung manuell bearbeiten kann.

Der große Vorteil von dispatch-conf ist, dass es die vom Paket mitgelieferten Konfigurationsdateien im Verzeichnis /etc/config-archive als *.dist (letzte Version) und *.dist.new (aktuelle Version) speichert; auch die tatsächlich im System vorliegende Version legt dispatch-conf bei jedem Lauf in diesem Verzeichnis ab. So entsteht automatisch eine Art Versionsverwaltung der Konfigurationsdateien.

Wer direkt eine richtige Versionsverwaltung für dispatch-conf verwenden möchte, kann dies auch in /etc/dispatch-conf.conf aktivieren:

```
use-rcs=yes
```

Dann muss man allerdings das Versionskontrollsystem app-text/rcs installieren, das nicht per Default zum System gehört:

```
gentoo ~ # emerge -av app-text/rcs

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-text/rcs-5.7-r3  330 kB

Total: 1 package (1 new), Size of downloads: 330 kB

Would you like to merge these packages? [Yes/No]
```

10.4 Besonderheiten und Probleme bei Update und Installation

Es gibt beim Aktualisieren einer Gentoo-Installation gelegentlich Situationen, die vom simplen, oben beschriebenen Update abweichen. Den häufigsten wollen wir uns hier widmen. Auch bei der Installation neuer Pakete gibt es einige Spezialfälle, die wir hier ebenfalls behandeln.

10.4.1 Slots

Wir haben im letzten Abschnitt gesehen, dass `emerge` Pakete, die ein Update benötigen, mit der Markierung

```
[ebuild     U ] ...
```

versieht. Stellenweise finden sich auch Pakete, die schon installiert sind, die `emerge` aber trotzdem bei einem Update als Neuinstallation markiert (siehe auch Seite 110). Dies ist z. B. beim Kernel der Fall:

```
emerge -av =sys-kernel/gentoo-sources-2.6.16-r13

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  NS   ] sys-kernel/gentoo-sources-2.6.16-r13  USE="-build -symli
nk (-ultra1)" 40,119 kB

Total: 1 package (1 in new slot), Size of downloads: 40,119 kB

Would you like to merge these packages? [Yes/No] No
```

Das `S` hinter dem `N` zeigt an, dass `emerge` die neue Version in einem neuen *Slot* installiert.

Im Normalfall ist es sinnvoll, nur eine einzelne Version eines Paketes installiert zu haben und bei einem Update die alte Version zu ersetzen. Bei einigen Programmen oder Bibliotheken ändert sich jedoch mit der Versionsnummer die Funktionalität signifikant, und es werden mehrere Versionen im System benötigt. Dies ist z. B. auch bei `automake` der Fall.

Wenn wir in der Datenbank installierter Pakete (in `/var/db/pkg`) nachschauen (eleganter geht dies mit `equery`, siehe Seite 246), sehen wir, dass `sys-devel/automake` in mehreren Versionen vorliegt:

```
gentoo ~ # ls /var/db/pkg/sys-devel/ | grep automake
automake-1.10
automake-1.9.6-r2
automake-wrapper-3-r1
```

Diese sind notwendig, da nicht alle Releases von `automake` miteinander kompatibel sind. Für das Kompilieren mancher Pakete sind spezifische Versionen von `automake` erforderlich. Um also zu gewährleisten, dass wir auf einem Gentoo-System auch alle Software erstellen können, sind die oben gelisteten Varianten alle im Standardsystem verfügbar.

Bei den meisten Paketen, die Portage in Slots installiert, ist es jedoch nicht notwendig, alle installierten Versionen auch wirklich im System zu behalten. Der Kernel wird, wie oben gezeigt, ebenfalls in Slots installiert. Allerdings dient dies nur dazu, dem Nutzer den Rückgriff auf verschiedene Versionen des Kernels zu erlauben. Da der Kernel das Herzstück des Systems darstellt, liegt es nahe, bei einem Kernel-Upgrade zuerst einmal die alte Version des Kernels zu behalten und diese erst zu entfernen, wenn eine neue Version auch wirklich stabil läuft.

Entsprechend installiert `emerge` den Source Code in verschiedenen Slots unter `/usr/src`. Sobald eine ältere Version nicht mehr notwendig ist, können wir diese bedenkenlos entfernen (was im Falle des Kernels auch durchaus Speicherplatz spart):

```
gentoo ~ # emerge -Cp =sys-kernel/gentoo-sources-2.6.16-r13
```

Das ist hier nur eine alte Beispielversion, die auf Ihrem System vermutlich nicht mehr vorhanden ist.

Haben sich mehrere alte Versionen angesammelt, hilft die Option `--prune` (bzw. `-P`), die alle Versionen bis auf die neueste entfernt. Diese Option ist allerdings nicht ungefährlich und wir sollten sie nur mit Vorsicht verwenden.

Häufig schlägt der erste Aufruf mit folgender Meldung fehl:

```
gentoo ~ # emerge --prune --pretend sys-devel/automake

Calculating dependencies... done!

Dependencies could not be completely resolved due to
the following required packages not being installed:
...
```

Bevor `--prune` Paketversionen zum Entfernen vorschlägt, prüft Portage zumindest alle bekannten Abhängigkeiten der aktuell installierten Pakete daraufhin, ob sie eine Version eines bestimmten Slots benötigen. Dieser Test schlägt schnell fehl, wenn die Maschine nicht auf dem neuesten Stand ist. Der Aufruf `emerge --update --newuse --deep world` bringt das System auf den neuesten Stand und sollte dieses Problem beseitigen.

Danach sollte der Aufruf folgendermaßen aussehen:

```
gentoo ~ # emerge --prune --pretend sys-devel/automake

>>> These are the packages that would be unmerged:

 sys-devel/automake
    selected: 1.10
   protected: 1.9.6-r2
     omitted: none

>>> 'Selected' packages are slated for removal.
>>> 'Protected' and 'omitted' packages will not be removed.
```

Hier ist `--prune` also der Meinung, dass derzeit kein Paket im System von `sys-devel/automake-1.10` abhängt. Alle anderen Versionen (hier `sys-devel/automake-1.9.6-r2`) werden noch benötigt.

Damit sollte man das Paket entfernen können. Bei dieser Operation ist dennoch Vorsicht geboten, da manche Pakete unabhängig von den für sie gelisteten Abhängigkeiten in der Lage sind, Bibliotheken automatisch zu ermitteln und einzubinden. Beim Kompilieren können solche Pakete quasi hinter dem Rücken von Portage automatisch die entsprechenden Verknüpfungen vornehmen.

Als Resultat sind diese Pakete dann nach der Deinstallation des eigentlich als überflüssig angenommenen Paketes nicht mehr brauchbar und wir müssen sie neu installieren.

Aus diesem Grund sollte man `--prune` nur für unkritische Pakete verwenden. Wir habe hier zwar `automake` als Beispiel verwendet, da es auf den meisten Systemen in mehreren Slots installiert ist, aber in der Praxis sollte man genau mit diesen systemkritischen Paketen nicht experimentieren.

10.4.2 Blocker

In seltenen Fällen kommt es beim Update oder der Installation zu Situationen, in denen sich zwei Pakete gegenseitig blockieren. Konstruieren wir einmal ein unwahrscheinliches, aber reproduzierbares Szenario, in dem wir versuchen, das Tool `app-admin/nologin` zu installieren:

```
gentoo ~ # emerge -pv app-admin/nologin

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-admin/nologin-20050522  3 kB
[blocks B     ] sys-apps/shadow (is blocking app-admin/nologin-20050522)
[blocks B     ] app-admin/nologin (is blocking sys-apps/shadow-4.0.18.1)

Total: 1 package (1 new, 2 blocks), Size of downloads: 3 kB
```

`emerge` meldet zurück, dass das derzeit installierte Paket `sys-apps/shadow` eine Installation von `app-admin/nologin-20050522` verhindert.

Führen wir `emerge app-admin/nologin` ohne das `-pv` aus, bricht `emerge` ab:

```
gentoo ~ # emerge app-admin/nologin
Calculating dependencies... done!

!!! Error: the sys-apps/shadow package conflicts with another package;
!!!        the two packages cannot be installed on the same system toget
her.
!!!        Please use 'emerge --pretend' to determine blockers.

For more information about Blocked Packages, please refer to the followi
ng
section of the Gentoo Linux x86 Handbook (architecture is irrelevant):

http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?full=1#blocked
```

Diese Konfliktsituation lässt sich lösen, indem wir das blockierende Paket (hier also `sys-apps/shadow`) über `emerge --unmerge` entfernen. Bevor man dies tut, sollte man sich allerdings darüber informieren, warum dieser Konflikt besteht, und sicher entscheiden können, ob ein Entfernen des blockierenden Paketes überhaupt die richtige Lösung darstellt.

Im gegebenen Fall würde das Entfernen von `sys-apps/shadow` zentrale Werkzeuge für das User-Management entfernen. `emerge` warnt vor dieser Aktion allerdings auch, da das `sys-apps/shadow`-Paket Bestandteil der `system`-Pakete ist:

```
gentoo ~ # emerge --unmerge sys-apps/shadow -pv

>>> These are the packages that would be unmerged:


!!! 'sys-apps/shadow' is part of your system profile.
!!! Unmerging it may be damaging to your system.


 sys-apps/shadow
    selected: 4.0.18.1
   protected: none
     omitted: none

>>> 'Selected' packages are slated for removal.
>>> 'Protected' and 'omitted' packages will not be removed.
```

Beschäftigt man sich eingehender mit der Ursache des Konflikts wird klar, dass `app-admin/nologin` ein BSD-spezifisches Tool ist und es auf unserem System nichts zu suchen hat. Mittlerweile ist das Paket sogar vollständig aus dem Portage-Baum verschwunden. Diese Demonstration ist also etwas konstruiert und im Normalfall sind Blockaden eher selten.

Aber wenn sie auftauchen, haben sie immer einen klar definierten Grund – und ohne diesen zu kennen, sollte man in keinem Fall die blockierenden Elemente aus dem System entfernen, sondern zunächst die Ursache des Problems identifizieren.

10.4.3 No-Fetch

Es gibt Softwarepakete, bei denen `emerge` die Quellen nicht direkt über FTP oder HTTP herunterladen kann, sondern die Intervention des Nutzers erfordert. Dies ist z. B. bei älteren Java-Paketen der Fall. Hier muss sich der Nutzer durch ein paar Webseiten klicken, bevor er an die Quell-Archive gelangt. `emerge` kann in solchen Fällen keine durchgehende Installation anbieten und bricht an der entsprechenden Stelle mit einem Hinweis auf die notwendigen Schritte zum Download der Quellen ab:

```
gentoo ~ # emerge =dev-java/sun-jdk-1.4*
...
>>> Emerging (6 of 6) dev-java/sun-jdk-1.4.2.13 to /

!!! dev-java/sun-jdk-1.4.2.13 has fetch restriction turned on.
!!! This probably means that this ebuild's files must be downloaded
!!! manually.  See the comments in the ebuild for more information.

 * Please download j2sdk-1_4_2_13-linux-i586.bin from:
 * http://javashoplm.sun.com/ECom/docs/Welcome.jsp?StoreId=22&PartDetail
```

```
Id=j2sdk-1.4.2_13-oth-JPR&SiteId=JSC&TransactionId=noreg
 * (first select 'Accept License', then click on 'self-extracting file'
 * under 'Linux Platform - Java(TM) 2 SDK, Standard Edition')
 * and move it to /usr/portage/distfiles
```

Sobald der Nutzer die Datei auf das lokale System heruntergeladen und in `/usr/portage/distfile` (bzw. `DISTDIR`, siehe Seite 143) platziert hat, kann `emerge` die Fetch-Phase überspringen und das Paket installieren.

Die Blockade des Downloads (*Fetch-Restriction*) zeigt `emerge` auch schon bei einem hypothetischen Lauf durch ein rotes `F` an:

```
gentoo ~ # emerge -pv =dev-java/sun-jdk-1.4*

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-portage/portage-utils-0.1.23  0 kB
[ebuild  N    ] app-arch/unzip-5.52-r1  0 kB
[ebuild  N    ] dev-java/java-config-wrapper-0.12-r1  0 kB
[ebuild  N    ] dev-java/java-config-2.0.31  0 kB
[ebuild  N    ] dev-java/java-config-1.3.7  0 kB
[ebuild  N F  ] dev-java/sun-jdk-1.4.2.13  USE="-X -alsa -doc -examples
-jce -nsplugin" 35,510 kB

Total: 6 packages (6 new), Size of downloads: 35,510 kB
Fetch Restriction: 1 package (1 unsatisfied)
```

10.4.4 Binäre Abhängigkeiten und revdep-rebuild

Wir haben schon in der Einleitung (siehe Seite 15) erwähnt, dass bei Gentoo die binären Abhängigkeiten erst auf dem Rechner des Benutzers entstehen. Portage nutzt diese Tatsache, um binäre Abhängigkeiten vollständig zu ignorieren und somit dem Benutzer eine sehr flexible Paketverwaltung anzubieten.

Das ändert allerdings nichts daran, dass diese binären Abhängigkeiten auf dem Rechner des Nutzers entstehen und wir sie verletzen können.

Ein einmal installiertes Paket ist durchaus von einer ganz bestimmten Version einer Bibliothek abhängig, nämlich derjenigen, die der Nutzer vorher installiert hat.

Schauen wir uns das am konkreten Apache-Beispiel an. Der Apache-Server hängt, sofern wir das `ssl`-USE-Flag aktiviert haben, von der Bibliothek `dev-libs/openssl` ab.

Etwas weiter oben (siehe Seite 220) haben wir auf eine niedrigere SSL-Version gewechselt, um das Aktualisieren von Konfigurationsdateien zu de-

monstrieren. Wir gehen an dieser Stelle davon aus, dass Sie derzeit also die Version `0.9.7l` installiert haben.

Zum Zeitpunkt unserer Apache-Installation war aber die Version `0.9.8d` installiert.

Versuchen wir im folgenden Schritt, den Apache-Server neu zu starten:

```
gentoo ~ # /etc/init.d/apache2 restart
 * Apache2 has detected a syntax error in your configuration files:
/usr/sbin/apache2: error while loading shared libraries: libssl.so.0.9.8
: cannot open shared object file: No such file or directory
```

Da der Apache bei der ursprünglichen Installation gegen die SSL-Bibliothek der Version `0.9.8` verlinkt wurde, haben wir mit dem Downgrade der Bibliothek die binäre Kompatibilität gebrochen. Das Beispiel ist natürlich eher konstruiert, da ein Downgrade eher selten stattfindet. Die Upgrade-Situation findet sich deutlich häufiger, führt aber zu demselben Problem. Würden wir jetzt die Version `openssl-0.9.7l` behalten wollen, kommen wir um eine Neuinstallation des Apache-Servers nicht umhin, um die binäre Kompatibilität wieder herzustellen.

Finden sich also nach einem Update Fehlermeldungen wie `error while loading shared libraries`, kann man davon ausgehen, dass man an irgendeiner Stelle die binäre Abhängigkeit gebrochen hat und die abhängigen Pakete neu installieren muss.

Nun wäre es sehr umständlich, diese Probleme manuell durch Testen der einzelnen Pakete zu lösen. Darum gibt es das Werkzeug `revdep-rebuild`, das den Prozess automatisiert. Das Tool stammt ebenfalls aus dem Paket `app-portage/gentoolkit`, das wir an dieser Stelle ja schon installiert haben sollten.

Da wir gerade bewusst die Abhängigkeit zwischen `net-www/apache` und `dev-libs/openssl` gebrochen haben, können wir `revdep-rebuild` einmal laufen lassen, um zu sehen, ob es uns das selbst generierte Problem meldet. Wir fügen die Option `--pretend` (bzw. `-p`) hinzu, um zu vermeiden, dass das Skript die Situation automatisch repariert:

```
gentoo ~ # revdep-rebuild -p
Configuring search environment for revdep-rebuild

Checking reverse dependencies...

Packages containing binaries and libraries broken by a package update
will be emerged.

Collecting system binaries and libraries... done.
  (/root/.revdep-rebuild.1_files)
```

```
Collecting complete LD_LIBRARY_PATH... done.
  (/root/.revdep-rebuild.2_ldpath)

Checking dynamic linking consistency...
  broken /usr/bin/ldapcompare (requires  libcrypto.so.0.9.8 libssl.so.0.
9.8)
  broken /usr/bin/ldapdelete (requires  libcrypto.so.0.9.8 libssl.so.0.9
.8)
...
  broken /usr/sbin/sshd (requires  libcrypto.so.0.9.8 libssl.so.0.9.8)
  broken /usr/sbin/ssmtp (requires  libcrypto.so.0.9.8 libssl.so.0.9.8)
 done.
  (/root/.revdep-rebuild.3_rebuild)

Assigning files to ebuilds... done.
  (/root/.revdep-rebuild.4_ebuilds)

Evaluating package order...
!!! Multiple versions within a single package slot have been
!!! pulled into the dependency graph:

('ebuild', '/', 'dev-db/mysql-5.0.26-r2', 'merge') (no parents)

('ebuild', '/', 'dev-db/mysql-5.0.38', 'merge') pulled in by
  ('ebuild', '/', 'dev-perl/DBD-mysql-3.0008', 'merge')
  ('ebuild', '/', 'virtual/mysql-5.0', 'merge')

 done.
  (/root/.revdep-rebuild.5_order)

All prepared. Starting rebuild...
emerge --oneshot -p =dev-lang/perl-5.8.8-r2 =net-misc/wget-1.10.2 =dev-l
ang/python-2.4.3-r4 =net-nds/openldap-2.3.30-r2 =dev-db/mysql-5.0.26-r2
=dev-perl/DBD-mysql-3.0008 =net-misc/openssh-4.5_p1-r1 =net-www/apache-2
.0.58-r2 =mail-mta/ssmtp-2.61-r2

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] dev-lang/perl-5.8.8-r2
[ebuild   R   ] net-misc/wget-1.10.2
[ebuild   R   ] dev-lang/python-2.4.3-r4
[ebuild   R   ] net-nds/openldap-2.3.30-r2
[ebuild   R   ] dev-db/mysql-5.0.26-r2
[ebuild   R   ] dev-perl/DBD-mysql-3.0008
[ebuild   R   ] net-misc/openssh-4.5_p1-r1  USE="ldap*"
[ebuild   R   ] net-www/apache-2.0.58-r2  USE="mpm-prefork* mpm-worker*
threads*"
[ebuild   R   ] mail-mta/ssmtp-2.61-r2
Now you can remove -p (or --pretend) from arguments and re-run revdep-re
build.
```

Zuletzt informiert uns `revdep-rebuild`, dass nicht nur `net-www/apache` nicht mehr funktioniert, sondern dass auch `dev-db/mysql` und noch einige weitere Pakete von OpenSSL abhängen und neu installiert werden müssten, wenn wir bei der älteren Version von `dev-libs/openssl` bleiben wollen.

Außerdem informiert `revdep-rebuild` darüber, dass nun ein zweiter Lauf ohne die Option `--pretend` angebracht sei. Würden wir diesen an dieser Stelle durchführen, würde `revdep-rebuild` die eigentliche Analyse der Situation nicht nochmals vornehmen, sondern das gespeicherte Resultat nutzen, um die defekten Pakete zu erneuern.

Wir wollen uns aber erst einmal die Funktionsweise des Werkzeugs genauer ansehen, damit klar wird, wie sich die binären Abhängigkeiten auswirken.

`revdep-rebuild` analysiert alle Dateien in einigen vorgegebenen Verzeichnissen, die üblicherweise kompilierte Programme enthalten. Die zu durchsuchenden Verzeichnisse gibt `revdep-rebuild` auch in der temporären Datei `~/.revdep-rebuild.0_env` aus (Variable `SEARCH_DIRS`). Alle ausführbaren Dateien in diesem Ordner finden sich dann in der temporären Datei `~/.revdep-rebuild.1_files` wieder. Die Orte, an denen Bibliotheken abgelegt sein können, finden sich in `~/.revdep-rebuild.2_ldpath`.

Vereinfacht ausgedrückt, nimmt `revdep-rebuild` dann für jedes binäre Programm eine Überprüfung mit `ldd` vor:

```
gentoo ~ # ldd /usr/sbin/apache2 | grep "not found"
        libssl.so.0.9.8 => not found
        libcrypto.so.0.9.8 => not found
```

`ldd` zeigt die Abhängigkeiten von Bibliotheken und damit eventuell verbundene Probleme an.

Alle Dateien mit solchen nicht erfüllten Abhängigkeiten speichert `revdep-rebuild` unter `~/.revdep-rebuild.3_rebuild` ab. Hier durchsucht das Programm dann die installierten Pakete, um festzustellen, zu welchen Paketen die defekten Programme gehören. Das Ergebnis der Analyse findet sich in `~/.revdep-rebuild.4_ebuilds` wieder und enthält im vorliegenden Fall:

```
gentoo ~ # cat ~/.revdep-rebuild.4_ebuilds
dev-perl/DBD-mysql-3.0008
dev-lang/python-2.4.3-r4
dev-lang/perl-5.8.8-r2
dev-db/mysql-5.0.26-r2
net-nds/openldap-2.3.30-r2
mail-mta/ssmtp-2.61-r2
net-www/apache-2.0.58-r2
net-misc/wget-1.10.2
net-misc/openssh-4.5_p1-r1
```

Diese ließen sich nun mit einem erneuten Aufruf von `revdep-rebuild` ohne die Option `--pretend` reparieren. In unserem Fall ist es aber einfacher, wieder die neueste Version von OpenSSL zu installieren und damit die zerstörten Abhängigkeiten zu reparieren:

```
gentoo ~ # emerge openssl
```

Danach sollte sich der Apache-Server wieder problemlos starten lassen:

```
gentoo ~ # /etc/init.d/apache2 restart
 * Starting apache2 ...             [ ok ]
```

Wie sich mit `equery` (siehe Kapitel 11.1.1) schon im Voraus überprüfen lässt, welche Pakete vom Update einer Bibliothek betroffen sein könnten, erklären wir genauer auf Seite 250.

10.4.5 Maskierte Pakete

Unter Umständen bricht Portage bei der Installation eines Paketes mit folgender Meldung ab:

```
gentoo ~ # emerge =sys-devel/gcc-3*
!!! All ebuilds that could satisfy "=sys-devel/gcc-3*" have been masked.
!!! One of the following masked packages is required to complete your re
quest:
- sys-devel/gcc-3.3.5.20050130-r1 (masked by: package.mask)
# These are release-specific things and won't show up in the tree this w
ay

- sys-devel/gcc-3.4.1-r3 (masked by: package.mask, ~x86 keyword)
- sys-devel/gcc-3.2.3-r4 (masked by: profile, package.mask)
- sys-devel/gcc-3.2.2 (masked by: profile, package.mask, missing keyword)
- sys-devel/gcc-3.3.6-r1 (masked by: package.mask)
- sys-devel/gcc-3.4.6-r1 (masked by: package.mask)
- sys-devel/gcc-3.4.6-r2 (masked by: package.mask)
- sys-devel/gcc-3.4.5-r1 (masked by: package.mask)
- sys-devel/gcc-3.4.4-r1 (masked by: package.mask)
- sys-devel/gcc-3.3.2-r7 (masked by: profile, package.mask)
- sys-devel/gcc-3.3.6 (masked by: package.mask)
- sys-devel/gcc-3.3.5-r1 (masked by: package.mask)
- sys-devel/gcc-3.1.1-r2 (masked by: profile, package.mask)
- sys-devel/gcc-3.4.5 (masked by: package.mask)
- sys-devel/gcc-3.4.6 (masked by: package.mask, ~x86 keyword)

For more information, see MASKED PACKAGES section in the emerge man page
or refer to the Gentoo Handbook.
```

In diesem Fall verweigert `emerge` die Installation je nach Version aus unterschiedlichen Gründen. So ist z. B. das Paket `sys-devel/gcc-3.4.6` als instabil markiert (`masked by: ~x86 keyword`). Die Blockade lässt sich aufheben, indem wir für dieses Paket das entsprechende instabile Keyword in der Datei `/etc/portage/package.keywords` hinzufügen (siehe 5.5).

Eine andere Fehlermeldung (`masked by: package.mask`) zeigt an, dass die Entwickler die gewünschte Version innerhalb der Datei `/usr/portage/profiles/package.mask` gesperrt haben. Üblicherweise wird ein Grund für die Maskierung angegeben, und man sollte als Nutzer im Normalfall davon absehen, die Version zu installieren. Die Blockade können wir jedoch, sofern wir uns sicher sind, über einen Eintrag in der Datei `/etc/portage/package.unmask` entfernen (siehe 5.5).

Auf x86-Maschinen zeigt Portage die Begründung `masked by: missing keyword` oder auch `masked by: -* keyword` und `masked by: -x86 keyword` selten an. Diese Fehler weisen darauf hin, dass das zu installierende Paket von den Entwicklern entweder noch nicht auf dieser Architektur getestet wurde (`missing keyword`) oder sie es als auf dieser Architektur nicht als nicht funktional markiert haben. Im Falle des fehlenden Keywords kann es trotzdem sein, dass das Paket problemlos funktioniert, es nur noch niemand getestet hat. In diesem Fall kann man die Entwickler über die Bug-Datenbank darüber informieren, dass man das Paket gerne auf der entsprechenden Architektur getestet sehen würde. Es hilft natürlich, wenn man selbst schon einmal das entsprechende Keyword zu dem Ebuild hinzufügt und das Paket selber testet (mehr dazu in Kapitel 15.1.1 auf Seite 323).

Als letzten möglichen Grund kann Portage `masked by: profile` anzeigen und dem Nutzer damit melden, dass das zu installierende Paket nicht mit dem gewählten Profil in Einklang zu bringen ist. Diese Schranke sollten wir nicht umgehen, da sie normalerweise triftige Gründe hat (z. B. läuft das Programm nur auf speziellen Rechner-Architekturen).

10.4.6 System-Pakete

Es war bereits von einer Warnung die Rede, die Portage ausgibt, wenn der Nutzer Pakete aus dem Basis-System zu entfernen beabsichtigt (siehe 10.4.2):

```
gentoo ~ # emerge --unmerge sys-apps/shadow -pv

>>> These are the packages that would be unmerged:

!!! 'sys-apps/shadow' is part of your system profile.
!!! Unmerging it may be damaging to your system.
```

Die Warnung sollten wir ernst nehmen, da die System-Pakete essentiell für das Funktionieren der Maschine sind und wir solche Pakete nie entfernen sollten!

10.4.7 Entwicklerfehler

Es gibt durchaus eine Reihe von Fehlern, bei denen der Nutzer unschuldig ist und die durch typische Entwicklerfehler zustande kommen. Dadurch, dass die Nutzer praktisch direkt am Portage-CVS-System hängen und alle Veränderungen am Originalbaum sofort über die Rsync-Mirror-Server weitergegeben werden, können schon kleine Ungenauigkeiten bei Veränderungen von Entwicklerseite zu Problemen beim Nutzer führen.

Aufgrund des typischen Ablaufs bei der Ebuild-Entwicklung (siehe Kapitel 15) kommt es hierbei vor allem zu den im Folgenden beschriebenen Fehlern. In allen Fällen sollte man einen entsprechenden Bug-Report in der Gentoo-Datenbank anlegen.

Falscher Digest

Alle Ebuilds und Quellarchive müssen im Portage-Baum über Checksummen identifizierbar sein. Diese erstellen die Entwickler bei Veränderungen am Ebuild und beim Einchecken neuer Versionen. Auf Nutzerseite verwendet Portage diese Checksummen, um die Integrität von Ebuilds und Quellen zu verifizieren. Damit kann der Nutzer sicher sein, dass er keinen Schadcode auf seinem Rechner ausführt, sofern er denn den Entwicklern ausreichend vertraut.

Es kann allerdings passieren, dass ein Entwickler beim Einchecken eines neuen Ebuilds oder der Veränderung eines alten Ebuilds vergisst, diese Checksummen zu aktualisieren. In diesem Fall scheitert Portage auf Nutzerseite mit der Meldung:

```
>>> checking ebuild checksums
!!! Digest verification failed:
```

Dieser Fehler ist in fast jedem Fall der Entwicklerseite zuzuschreiben. Wenn der Fehler einige Zeit später nach einem erneuten `emerge --sync` nicht verschwindet, sollte man der Bug-Datenbank einen entsprechenden Fehlerbericht hinzufügen.

Fehlende und zirkuläre Abhängigkeiten

Sind in einem Ebuild Abhängigkeiten definiert, die als Paket gar nicht existieren, so scheitert Portage ebenfalls:

```
emerge: there are no ebuilds to satisfy ">=sys-foo/depfoo-1".

!!! Problem with ebuild sys-foo/foo-1
!!! Possibly a DEPEND/*DEPEND problem.
```

Gleiches gilt, wenn in einem Ebuild Abhängigkeiten definiert sind, die in einem geschlossen Kreis resultieren:

```
!!! Error: circular dependencies:

ebuild / sys-foo/foo-1 depends on ebuild / sys-foo/foo-2
ebuild / sys-foo/foo-2 depends on ebuild / sys-foo/foo-1
```

Tritt einer dieser Fehler bei einem Paket des Portage-Baums auf, so ist dies ein klarer Fehler eines Entwicklers und wir sollten das Problem unter `http://bugs.gentoo.org` eintragen. Diese Fehler sind allerdings selten und treten eher auf, wenn man eigene Ebuilds entwickelt (siehe Kapitel 15).

10.5 System aufräumen

Wenn wir Pakete deinstallieren, entfernt `emerge` die bei der Installation ebenfalls installierten Abhängigkeiten nicht automatisch mit (siehe Seite 107). Genauso werden auch bei einer Aktualisierung nicht einfach Pakete deinstalliert, wenn die neue Version im Gegensatz zur vorigen nicht mehr von diesen abhängt.

Folglich sammelt unser System zwangsläufig mit der Zeit ungenutzte Bibliotheken an. Wer eine Software nur kurz testet, dann für uninteressant befindet und wieder deinstalliert, behält unweigerlich alle aufgrund von Abhängigkeiten installierten Pakete im System.

Für die gelegentliche Aufräumaktion bietet sich die Option `--depclean` an, die wir in jedem Fall erst einmal mit `--pretend` laufen lassen müssen:

```
gentoo ~ # emerge --depclean -p
*** WARNING *** Depclean may break link level dependencies. Thus, itis
*** WARNING *** recommended to use a tool such as ‘revdep-rebuild‘ (from
*** WARNING *** app-portage/gentoolkit) in order to detect such breakage.
*** WARNING ***
*** WARNING *** Also study the list of packages to be cleaned for any obv
ious
*** WARNING *** mistakes. Packages that are part of the world set will al
ways
*** WARNING *** be kept.  They can be manually added to this set with
*** WARNING *** ‘emerge --noreplace <atom>‘.  Packages that are listed in
*** WARNING ***  package.provided (see portage(5)) will be removed by
```

```
*** WARNING ***  depclean, even if they are part of the world set.
*** WARNING ***
*** WARNING ***  As a safety measure, depclean will not remove any packag
es
*** WARNING ***  unless *all* required dependencies have been resolved.
As a
*** WARNING ***  consequence, it is often necessary to run
*** WARNING ***  'emerge --update --newuse --deep world' prior to depclea
n.
...
```

Portage brüllt uns hier geradezu an, diese Option nur mit extremer Vorsicht zu verwenden. Das ist auch der Grund, warum dieser Vorgang nicht automatisch abläuft.

Der korrekte Ablauf für einen solchen Aufräumvorgang besteht also zunächst einmal in einer vollständigen Aktualisierung des Systems mit

```
gentoo ~ # emerge --update --newuse --deep world
```

Danach sollte man `--depclean` erneut mit `--pretend` laufen lassen und sich sehr genau ansehen, welche Pakete `emerge` dabei als überflüssig markiert.

Alle Pakete, bei denen man auch nur einen Hauch von Unsicherheit verspürt, ob sie nicht vielleicht wichtig für das System sind, sollte man über folgenden Befehl zu `world` hinzufügen:

```
gentoo ~ # emerge --noreplace PAKET
```

Diese Pakete sollte `emerge` nun bei einem erneuten Lauf von `--depclean` nicht mehr anzeigen.

So abgesichert, kann man anschließend die Option `--pretend` entfernen und die Aufräumaktion anlaufen lassen.

Zur Sicherheit überprüft man danach nochmals alle binären Abhängigkeiten mit `revdep-rebuild`.

10.6 Update-Zyklus

Da der Portage-Baum im CVS-System der Entwickler kontinuierlich auf die Rsync-Server gespiegelt wird, könnte man quasi stündlich das eigene Gentoo-System aktualisieren.

Man könnte, aber man sollte natürlich nicht.

Grundsätzlich haben die Mirror-Server die Policy, dass kein Benutzer mehrmals an einem Tag von einer IP-Adresse aus synchronisieren sollte. Im Normalfall dürfte dies auch wirklich nicht notwendig sein.

Es stellt sich darum die Frage nach einem sinnvollen Update-Zyklus. Was sind die ausschlaggebenden Punkte für die Entscheidung, wie oft man synchronisiert?

Fehler
: Die Tatsache, dass Entwickler auch nur Menschen sind, führt gelegentlich dazu, dass neuere Paket-Versionen Probleme mit sich bringen. Diese werden meist relativ schnell wieder korrigiert, da andere Benutzer vom gleichen Fehler betroffen sind. Nachdem die Entwickler den Fehler korrigiert haben, stellen sie meist eine neue Version zur Verfügung, die das Problem behebt.

  Beim automatischen Update mit hoher Frequenz erwischt man das Paket also erst einmal in der problematischen Fassung und muss es dann ein zweites Mal kompilieren und installieren.

Features
: Ein Versionssprung bei einer Software heißt nicht immer, dass die Entwickler kritische Fehler behoben haben oder dass neue Eigenschaften, die für uns wichtig sind, verfügbar sind.

  Wenn wir mit hoher Frequenz aktualisieren, kompilieren wir die Pakete bei jedem noch so kleinen Versionssprung neu.

Sicherheit
: Pakete, die als unsicher erkannt wurden, müssen wir so schnell wie möglich ersetzen, damit die eigene Maschine nicht kompromittiert werden kann.

  Eine höhere Frequenz der Aktualisierungen bedeutet also höhere Sicherheit.

Kompatibilität
: Je seltener wir aktualisieren, desto mehr weichen wir vom Stand des Portage-Baums ab. Nach längerer Zeit kann es passieren, dass Neuerungen nicht mehr direkt mit unserer Maschine kompatibel sind. Eine hohe Update-Frequenz vermeidet solche Probleme.

Im Prinzip stehen sich also der Aufwand beim Kompilieren und die Aktualität des Systems gegenüber.

Es gibt Nutzer, die den Portage-Baum jede Nacht synchronisieren und neue Pakete automatisch über einen Cron-Job kompilieren und installieren. Das Problem dabei: Das automatische Update kann die Software zwar aktualisieren und sollte im Normalfall auch fehlerfrei durchlaufen, aber ab und an besteht mit einem Update einer Software auch die Notwendigkeit, die Konfiguration des Paketes oder den von der Software verwalteten Datenbestand zu aktualisieren. Diese Prozesse sind im Regelfall nicht durch den

Ebuild automatisiert, und es besteht die Chance, dass die neuen Pakete in einem nicht funktionierenden Zustand enden.

Dennoch: Wer gerne an der vordersten Front der Software-Entwicklung steht, der findet die notwendige Konfiguration für diese Art der nächtlichen Aktualisierung in Kapitel 16.6.4.

Generell ist der interaktive Modus, bei dem wir die Anweisungen am Ende einer Installation befolgen, aber vorzuziehen. Auch ist ein tägliches Update der Pakete nicht wirklich zu empfehlen, denn schließlich bringt es wenig, Pakete zu aktualisieren, wenn der einzige ersichtliche „Vorteil“ in einer erhöhten Versionsnummer besteht.

Die Aktualisierung einmal im Monat ist meist sinnvoller. Als Zeitpunkt eignet sich ein Moment, in dem notfalls noch die Zeit vorhanden ist, das aktualisierte System ein wenig zu testen und eventuell auftretende Probleme zu korrigieren.

Allerdings ist Sicherheit dabei ein problematischer Punkt. Eine Zeitspanne von einem Monat kann bei gravierenden Lücken durchaus problematisch sein. Der Sicherheitscheck lässt sich unter Gentoo allerdings recht einfach automatisieren, so dass wir einmal täglich überprüfen können, ob Pakete mit Sicherheitslücken installiert sind. Die entsprechende Konfiguration beschreiben wir in Kapitel 16.6.4. Finden sich solche Lecks, können wir die betroffene Software gezielt aktualisieren.

Damit reicht es dann sogar aus, halbjährlich ein Update auf den Gentoo-Maschinen durchzuführen. So kommen zwar meist einige hundert zu aktualisierende Pakete zusammen, aber der Aufwand ist geringer als die zusammengenommene Zeit, die man in das tägliche Update investieren muss.

Kapitel

11 Tools

Bevor wir uns nun anschicken, Anwendungssoftware zu installieren und die Maschine ihrer eigentlichen Bestimmung zuzuführen, schauen wir uns einige Werkzeuge an, die speziell für Gentoo entwickelt wurden und den täglichen Umgang mit dem System erleichtern.

11.1 Das Paket app-portage/gentoolkit

Dieses Paket haben wir bereits für die Arbeit mit `euse` installiert (siehe 5.1.1). Auch `revdep-rebuild` stammt aus diesem Paket (siehe 10.4.4). Es enthält darüber hinaus vier weitere Skripte für die Administration eines Gentoo-Systems: `equery`, `glsa-check`, `eclean` und `eread`.

Wir wollen uns diese Werkzeuge im Folgenden genauer anschauen und auf das vielseitige `equery` sowie das Skript `glsa-check` das Hauptaugenmerk legen. `equery` liefert Zusatzinformationen zum Paketmanagement, und `glsa-check` ist für das Testen des eigenen Systems auf Sicherheitslücken ein unverzichtbares Werkzeug.

11.1.1 equery

`equery` ergänzt das Paketmanagement und liefert wichtige Paketinformationen, wie z. B. eine Liste der Dateien, die zu einem Paket gehören. Um die Ausgabe kurz zu halten, fragen wir an dieser Stelle einmal, welche Dateien das kleine Paket `sys-apps/which` installiert, und übergeben `equery` die Aktion `files`, gefolgt vom Paketnamen:

```
gentoo ~ # equery files sys-apps/which
[ Searching for packages matching sys-apps/which... ]
* Contents of sys-apps/which-2.16:
/usr
/usr/bin
/usr/bin/which
/usr/share
/usr/share/doc
/usr/share/doc/which-2.16
/usr/share/doc/which-2.16/AUTHORS.bz2
/usr/share/doc/which-2.16/EXAMPLES.bz2
/usr/share/doc/which-2.16/NEWS.bz2
/usr/share/doc/which-2.16/README.alias.bz2
/usr/share/doc/which-2.16/README.bz2
/usr/share/info
/usr/share/info/which.info.bz2
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/which.1.bz2
```

Die Aktion `files`, kombiniert mit der Option `--filter`, zeigt z. B., welche Tools das Paket `app-portage/gentoolkit` eigentlich installiert:

```
gentoo ~ # equery files --filter=cmd app-portage/gentoolkit
[ Searching for packages matching app-portage/gentoolkit... ]
* Contents of app-portage/gentoolkit-0.2.3:
/usr/bin/eclean
/usr/bin/eclean-dist -> eclean
/usr/bin/eclean-pkg -> eclean
/usr/bin/equery
/usr/bin/eread
/usr/bin/euse
/usr/bin/glsa-check
/usr/bin/revdep-rebuild
```

Die Filter-Funktion erlaubt hier mit der Auswahl `cmd` die Selektion von Dateien, die innerhalb der Variablen `PATH` liegen (also Dateien, die in einem der Verzeichnisse liegen, die das System nach ausführbaren Dateien durchsucht). Die Filter-Funktion bietet einige weitere Optionen, die die Man-Page beschreibt.

Häufig stellt sich die Frage, welches Paket eigentlich welche Datei installiert hat. Nutzt man z. B. das Tool `equery`, weiß aber eigentlich gar nicht, woher es stammt, kann man es selbst über die Aktion `belongs` gefolgt vom Dateipfad befragen:

```
gentoo ~ # which equery
/usr/bin/equery
gentoo ~ # equery belongs 'which equery'
[ Searching for file(s) /usr/bin/equery in *... ]
app-portage/gentoolkit-0.2.3 (/usr/bin/equery)
```

Da eine Datei im Normalfall nur zu einem Paket gehört, ist es sinnvoll, die Aktion `belongs` mit der Option `-e` zu versehen. `equery` bricht dann ab, sobald es die erste Übereinstimmung gefunden hat, und wühlt sich nicht noch durch die übrige Paketliste.

Gelegentlich findet sich in den Gentoo-Foren auch die Frage, in welchem Paket sich denn das Programm `xyz` befindet. Mancher hat ein Programm auf einer anderen Linux-Variante lieb gewonnen, findet sich aber nicht gleich im Portage-Baum zurecht und würde gerne nach dem Dateinamen suchen, um das passende Paket zu ermitteln.

Unter Gentoo lassen sich – im Gegensatz zu RPM-basierten Distributionen – nicht installierte Pakete leider nicht durchsuchen. Unter Gentoo wird jedes Paket nur durch den Ebuild repräsentiert, und dieser ist schließlich kein gepacktes Archiv zu installierender Dateien, sondern eine allgemeine Installationsanweisung. Der Ebuild lädt das Quellpaket selbst erst herunter, und die meisten Programmdateien erstellt Portage erst während des Kompilierens. Es kann bei diesem Verfahren folglich kein „Vorwissen“ über die zu installierenden Dateien geben.

Daher sind unter Gentoo gewisse Umwege notwendig:

- Kennt man das gesuchte Werkzeug von einer anderen Linux-Distribution, sollte man überprüfen, aus welchem Paket das Programm dort stammt. In vielen Fällen findet man unter Gentoo ein Paket gleichen Namens.
- Die Suche im Internet sollte normalerweise den Weg zu dem Software-Projekt weisen, das ein Programm mit dem gesuchten Namen anbietet. Das passende Gentoo-Paket sollte sich dann mit den in Kapitel 13 beschriebenen Methoden finden lassen.
- Als letzter, wenn auch vielleicht nicht schnellster Rettungsanker bietet sich eine Anfrage in einem Gentoo-Forum an.

Unter 10.4.1 wollten wir bereits prüfen, welche Versionen eines Paketes installiert sind. Auch wenn sich dazu die interne Portage-Datenbank unter `/var/db/pkg` befragen lässt, bietet `equery list` einen eleganteren Weg, der auch eine übersichtlichere Ausgabe produziert:

```
gentoo ~ # equery list sys-devel/automake
[ Searching for package 'automake' in 'sys-devel' among: ]
 * installed packages
[I--] [  ] sys-devel/automake-1.9.6-r2 (1.9)
[I--] [  ] sys-devel/automake-1.10 (1.10)
[I--] [  ] sys-devel/automake-wrapper-3-r1 (0)
```

Das initiale `I` markiert, dass das entsprechende Paket installiert ist. Erweitert man den Befehl um die Option `-p` durchsucht `equery` den kompletten Portage-Baum. Diese Operation ist jedoch ähnlich langsam wie `emerge --search` (siehe Kapitel 13.1), und wieder ist der Befehl `esearch` vorzuziehen:

```
gentoo ~ # equery list -p sys-devel/automake
[ Searching for package 'automake' in 'sys-devel' among: ]
 * installed packages
[I--] [  ] sys-devel/automake-1.9.6-r2 (1.9)
[I--] [  ] sys-devel/automake-1.10 (1.10)
[I--] [  ] sys-devel/automake-wrapper-3-r1 (0)
 * Portage tree (/usr/portage)
[-P-] [  ] sys-devel/automake-1.4_p6 (1.4)
[-P-] [  ] sys-devel/automake-1.5 (1.5)
[-P-] [  ] sys-devel/automake-1.6.3 (1.6)
[-P-] [  ] sys-devel/automake-1.7.9-r1 (1.7)
[-P-] [  ] sys-devel/automake-1.8.5-r3 (1.8)
[-P-] [  ] sys-devel/automake-wrapper-1-r1 (0)
[-P-] [  ] sys-devel/automake-wrapper-2-r1 (0)
```

Der Output zeigt in knapper Form, dass die entsprechenden Pakete nicht installiert, sondern nur innerhalb von Portage definiert sind (fehlendes `I`, Markierung `P`). Einige Ebuilds sind auch als instabil maskiert (`M~`).

Geht einmal der Speicherplatz der Festplatte zur Neige, stellt sich schnell die Frage, welche Pakete wie viel Platz benötigen, um mögliche Kandidaten zum Löschen zu identifizieren. Die Antwort liefert der Aufruf `equery size`:

```
gentoo ~ # equery size net-www/apache
[ Searching for packages matching net-www/apache... ]
* size of net-www/apache-2.0.58-r2
           Total files : 492
           Total size  : 2721.27 KiB
```

Treten Probleme bei einer Software auf, die vorher völlig unproblematisch lief, könnte eine mögliche Ursache darin liegen, dass man unabsichtlich Dateien des zugehörigen Paketes modifiziert hat. In dem Fall kann es helfen, das gesamte Paket neu zu installieren und damit die Dateien wieder in ihren Ursprungszustand zu versetzen.

Portage speichert die Checksummen aller installierten Dateien eines Paketes, und `equery` kann problemlos prüfen, ob sich das Paket noch im Originalzustand befindet.

Beschädigen wir einmal mutwillig das Paket `sys-apps/which` und sehen uns die Ausgabe von `equery check` an:

```
gentoo ~ # mv /usr/share/doc/which-2.16/AUTHORS.bz2 /tmp/
gentoo ~ # mv /usr/share/doc/which-2.16/EXAMPLES.bz2 /tmp/
gentoo ~ # echo "hello world" > /usr/share/doc/which-2.16/EXAMPLES.bz2
gentoo ~ # equery check sys-apps/which
[ Checking sys-apps/which-2.16 ]
!!! /usr/share/doc/which-2.16/EXAMPLES.bz2 has incorrect md5sum
!!! /usr/share/doc/which-2.16/AUTHORS.bz2 does not exist
 * 14 out of 16 files good
```

Entsprechend zeigt `equery` die fehlende und die modifizierte Datei als verändert an.

Wenn wir die Veränderungen rückgängig machen, beruhigt sich `equery` wieder:

```
gentoo ~ # mv /tmp/EXAMPLES.bz2 /usr/share/doc/which-2.16/
gentoo ~ # mv /tmp/AUTHORS.bz2 /usr/share/doc/which-2.16/
gentoo ~ # equery check which
[ Checking sys-apps/which-2.16 ]
 * 16 out of 16 files good
```

`equery` bietet zudem einige paketorientierte Funktionen in Bezug auf USE-Flags. So lassen sich über `equery uses` die Flags anzeigen, die für ein bestimmtes Paket gerade aktiviert sind, bzw. die Flags, mit denen wir das Paket installiert haben:

```
gentoo ~ # equery uses net-www/apache
[ Searching for packages matching net-www/apache... ]
[ Colour Code : set unset ]
[ Legend : Left column  (U) - USE flags from make.conf              ]
[        : Right column (I) - USE flags packages was installed with ]
[ Found these USE variables for net-www/apache-2.0.58-r2 ]
 U I
 + + apache2        : Chooses Apache2 support when a package supports bo
th Apache1 and Apache2
 - - debug          : Enable extra debug codepaths, like asserts and ext
ra output. If you want to get meaningful backtraces see http://www.gento
o.org/proj/en/qa/backtraces.xml .
 - - doc            : Adds extra documentation (API, Javadoc, etc)
 + + ldap           : Adds LDAP support (Lightweight Directory Access Pr
otocol)
 - - mpm-itk        : (experimental) Itk MPM - child processes have sepe
rate user/group ids
 - - mpm-leader     : (experimental) Leader MPM - leaders/followers vari
ent of worker MPM
 - - mpm-peruser    : (experimental) Peruser MPM - child processes have
seperate user/group ids
```

```
 + - mpm-prefork    : Prefork MPM - non-threaded, forking MPM - similiar
 manner to Apache 1.3
 - - mpm-threadpool : (experimental) Threadpool MPM - keeps pool of idle
 threads to handle requests
 + - mpm-worker     : Worker MPM - hybrid multi-process multi-thread MPM
 - - selinux        : !!internal use only!! Security Enhanced Linux supp
ort, this must be set by the selinux profile or breakage will occur
 + + ssl            : Adds support for Secure Socket Layer connections
 - - static-modules : Build modules into apache instead of having them l
oad at run time
 + - threads        : Adds threads support for various packages. Usually
 pthreads
```

Möchte man ein USE-Flag systemweit aktivieren oder deaktivieren, sollte man sich zuvor informieren, auf welche installierten Pakete dies überhaupt Einfluss haben kann. Diese Funktionalität bietet `equery hasuse`:

```
gentoo ~ # equery hasuse ssl
[ Searching for USE flag ssl in all categories among: ]
 * installed packages
[I--] [  ] dev-lang/python-2.4.3-r4 (2.4)
[I--] [  ] mail-mta/ssmtp-2.61-r2 (0)
[I--] [  ] net-nds/openldap-2.3.30-r2 (0)
[I--] [  ] dev-db/mysql-5.0.26-r2 (0)
[I--] [  ] net-misc/wget-1.10.2 (0)
[I--] [  ] net-www/apache-2.0.58-r2 (2)
```

Dieser Test gilt allerdings nur für tatsächlich installierte Pakete und zeigt nicht sämtliche Pakete im Portage-Baum an, die das angegebene USE-Flag unterstützen. Dafür lässt sich die Aktion `hasuse` wieder mit der Option `-p` versehen, wodurch `equery` den gesamten Portage-Baum nach USE-Flags durchsucht, was aber die Operation stark verlangsamt. Für eine beschleunigte Variante, siehe Kapitel 11.2.

Um Situationen vorzubeugen, in denen wir, wie in Kapitel 10.4.4 beschrieben, Pakete beschädigen, indem wir eine wichtige Bibliothek neu installieren, können wir bei entsprechenden Änderungen im Voraus überprüfen, welche Pakete überhaupt von der Bibliothek abhängen und so in Mitleidenschaft gezogen werden könnten. Hier hilft die Aktion `depends` des Befehls `equery`:

```
gentoo ~ # equery depends dev-libs/openssl
[ Searching for packages depending on dev-libs/openssl... ]
app-misc/ca-certificates-20061027.2 (dev-libs/openssl)
dev-db/mysql-5.0.26-r2 (ssl? >=dev-libs/openssl-0.9.6d)
dev-lang/python-2.4.3-r4 (!build & ssl? dev-libs/openssl)
mail-mta/ssmtp-2.61-r2 (ssl? dev-libs/openssl)
net-misc/openssh-4.5_p1-r1 (>=dev-libs/openssl-0.9.6d)
net-misc/wget-1.10.2 (ssl? >=dev-libs/openssl-0.9.6b)
```

```
net-nds/openldap-2.3.30-r2 (!minimal & samba? dev-libs/openssl)
                           (!minimal&smbkrb5passwd? dev-libs/openssl)
                           (ssl? dev-libs/openssl)
net-www/apache-2.0.58-r2 (ssl? dev-libs/openssl)
```

Wir bekommen alle derzeit installierten Pakete genannt, die von OpenSSL abhängen. Entsprechende Checks sind auch dann sinnvoll, wenn man ein Paket aus dem System entfernen möchte und nicht sicher ist, ob man damit andere Pakete beschädigt.

Umgekehrt lassen sich die Abhängigkeiten eines Paketes über die Aktion `depgraph` ausgeben. Wir limitieren hier die Ausgabe etwas, indem wir `grep` verwenden, um einige überflüssige Informationen zu verstecken:

```
gentoo ~ # equery depgraph -U sys-apps/which | grep -v "^\["
sys-apps/which-2.16:
‘-- sys-apps/which-2.16
 ‘-- sys-apps/texinfo-4.8-r5
  ‘-- sys-libs/ncurses-5.5-r3
   ‘-- sys-libs/gpm-1.20.1-r5
  ‘-- virtual/libintl-0 (virtual/libintl)
   ‘-- sys-devel/gettext-0.16.1
    ‘-- virtual/libiconv-0 (virtual/libiconv)
    ‘-- dev-libs/expat-1.95.8
```

Damit haben wir die Funktionalität von `equery` ausreichend beschrieben. Es bleibt noch anzumerken, dass dieses Skript auch einen Nachteil hat: Es ist bei manchen Operationen leider recht langsam. Das hat einige Entwickler so sehr gestört, dass sie einen Teil der Funktionen von `equery` in C neu geschrieben haben. Dieses sehr schnelle Set an Programmen ist mit dem Paket `app-portage/portage-utils` erhältlich. Wir besprechen es etwas weiter unten in Abschnitt 11.2.

11.1.2 Gentoo-Sicherheit und glsa-check

Das zweite wichtige Werkzeug des Paketes `app-portage/gentoolkit`, das wir hier besprechen wollen heißt `glsa-check` und fokussiert sich auf die Sicherheit unseres Systems.

Für die Sicherheit der Distribution ist das *Gentoo Linux Security Project* – ein eigenes Gentoo-Unterprojekt – zuständig. Erster Anlaufpunkt in Bezug auf Sicherheitsfragen ist die Projektseite.[1] Die Hauptaufgabe des Teams besteht darin, Sicherheitsprobleme möglichst zeitnah zu beurteilen und eventuell ein sogenanntes *Gentoo Linux Security Advisory* (GLSA) herauszugeben.

[1] `http://www.gentoo.org/proj/en/security/`

Gentoo Linux Security Advisories

Ein GLSA beschreibt das Problem, die betroffenen Software-Versionen, die möglichen Auswirkungen und vor allem auch die notwendigen Aktionen, um das Problem zu beseitigen. Sie stellen damit die wichtigste Informationsquelle dar, wenn man an einem sicheren System interessiert ist.

GLSAs können über die Mailing-Liste `gentoo-announce`[2] bezogen werden und stehen auch als RSS-Feed zur Verfügung.[3] Eine vollständige und kontinuierlich aktualisierte Liste an GLSAs gibt es ebenfalls.[4]

Natürlich ist Sicherheit für jede Distribution ein essentielles Thema. Die aktive Verfolgung von Sicherheitslücken kostet aber auch eine nicht unerhebliche Menge Zeit. Um das Sicherheitsteam also nicht über Gebühr zu belasten, wird nicht gleich jede Rechnerarchitektur und jeder Ebuild gleichermaßen betreut.

Die derzeit gültigen Richtlinien für das Sicherheitsteam finden sich auf der Webseite.[5] Wir wollen hier nur die wichtigsten Aspekte beleuchten.

Derzeit werden folgende Architekturen unterstützt:

- `x86`
- `ppc`
- `sparc`
- `amd64`
- `alpha`
- `ppc64`
- `hppa`

Es ist allerdings nicht so, dass Architekturen wie `arm` oder `mips`, die in dieser Liste nicht aufgeführt sind, keinerlei Sicherheitsupdates erfahren würden. Ein Paketupdate auf den unterstützten Architekturen führt natürlich auch zu einem Update auf den Architekturen, die offiziell nicht vom Sicherheitsteam unterstützt werden.

Allerdings bestünde für das Sicherheitsteam z. B. keine Verpflichtung, ein GLSA herauszugeben, wenn es eine `mips`-spezifische Sicherheitslücke gäbe. Auf diesen Architekturen ist man also selbst stärker in der Pflicht, sich um die Sicherheit zu kümmern.

[2] `http://www.gentoo.org/main/en/lists.xml`
[3] `http://www.gentoo.org/rdf/en/glsa-index.rdf`
[4] `http://www.gentoo.org/security/en/glsa/index.xml`
[5] `http://www.gentoo.org/security/en/vulnerability-policy.xml`

In Bezug auf die einzelnen Ebuilds ist das entscheidende Kriterium, ob der Ebuild als stabil markiert ist oder nicht. Wenn ein Paket auf keiner Architektur als stabil markiert ist oder keine als stabil markierte Version von der Sicherheitslücke betroffen ist, wird *kein* GLSA herausgegeben. Damit ist der Benutzer für die Sicherheit jedes einzelnen instabilen Paketes, das er installiert, selbst verantwortlich. Ein nicht ganz unwichtiger Aspekt, den man bedenken sollte, bevor man eine instabile Version wählt.

Gibt es stabile Versionen, dann richtet sich die zu erwartende Reaktionszeit im Wesentlichen nach der Schwere der Sicherheitslücke und der Zahl der Nutzer eines Paketes. Bei weit verbreiteten Paketen wird für kritische Probleme, über die ein externer Angreifer z. B. `root`-Rechte erhalten könnte, ein Zeitrahmen von ein bis drei Tagen vorgegeben. Diese Art von Fehlern sind glücklicherweise sehr selten.

Die maximale Reaktionszeit bei wenig verbreiteten Paketen und geringfügigen Sicherheitsproblemen (z. B. *Cross Site Scripting* bei Web-Anwendungen) liegt bei 40 Tagen. Bei diesen weniger relevanten Fällen ist das Sicherheitsteam auch nicht verpflichtet, ein GLSA herauszugeben. Im Normalfall wird unter den Mitgliedern des Teams abgestimmt, ob sie es für erforderlich halten oder nicht.

Kommen wir jetzt zu einem Werkzeug, das unser Sicherheitsmanagement sehr vereinfacht, indem es uns erlaubt, unser System automatisch auf die veröffentlichen GLSAs hin zu überprüfen.

glsa-check

`glsa-check` überprüft unser System automatisiert auf unsichere Pakete und ist damit ein unverzichtbares Skript, um das System sicher zu halten. Es erlaubt die schnelle Identifizierung aller Pakete mit Sicherheitslücken. Diese Sichertsheitsinformationen sind für den Systemadministrator essentiell, da sie die Pakete definieren, die in jedem Fall ein Update erfahren sollten, um die Wahrscheinlichkeit für einen erfolgreichen Angriff von außen so gering wie möglich zu halten.

Der Befehl `glsa-check --list` gleicht jedes herausgegebene GLSA mit den auf dem System installierten Paketen ab und listet alle Advisories mit einem entsprechenden Code (`[U]`: nicht betroffen, `[N]`: betroffen). Auf folgende Weise lassen sich alle Pakete mit Sicherheitsproblemen und deren Kurzzusammenfassung anzeigen:

```
gentoo ~ # glsa-check --list | grep "\[N\]"
[A] means this GLSA was already applied,
[U] means the system is not affected and
[N] indicates that the system might be affected.

200611-06 [N] OpenSSH: Multiple Denial of Service vulnerabilities ( net-
```

```
misc/openssh )
200609-13 [N] gzip: Multiple vulnerabilities ( app-arch/gzip )
200609-17 [N] OpenSSH: Denial of Service ( net-misc/openssh )
200610-07 [N] Python: Buffer Overflow ( dev-lang/python )
```

Die gleiche Information, aber reduziert auf die ID der GLSA, liefert `glsa-check` mit der Kombination `glsa-check --test all` (bzw. `-t all`).

```
gentoo ~ # glsa-check -t all
This system is affected by the following GLSAs:
200611-06
200609-13
200609-17
200610-07
```

Um detailliertere Informationen zu einem Problem zu bekommen, lässt sich `glsa-check` mit der Option `--dump` verwenden:

```
gentoo ~ # glsa-check --dump 200610-07
                 GLSA 200610-07:
Python: Buffer Overflow
================================================================================
Synopsis:          A buffer overflow in Python's "repr()" function can be
                   exploited to cause a Denial of Service and potentially
                   allows the execution of arbitrary code.
Announced on:      October 17, 2006
Last revised on:   October 17, 2006: 02

Affected package:  dev-lang/python
Affected archs:    All
Vulnerable:        <2.4.3-r4
Unaffected:        >=2.4.3-r4 >=~2.3.5-r3


Related bugs:      149065

Background:        Python is an interpreted, interactive,
                   object-oriented, cross-platform programming language.

Description:       Benjamin C. Wiley Sittler discovered a buffer overflow
                   in Python's "repr()" function when handling
                   UTF-32/UCS-4 encoded strings.

Impact:            If a Python application processes attacker-supplied
                   data with the "repr()" function, this could
                   potentially lead to the execution of arbitrary code
                   with the privileges of the affected application or a
                   Denial of Service.

Workaround:        There is no known workaround at this time.
```

```
Resolution:        All Python users should update to the latest version:

                   # emerge --sync
                   # emerge --ask --oneshot --verbose
                   ">=dev-lang/python-2.4.3-r4"

References:
                   CVE-2006-4980: http://cve.mitre.org/cgi-bin/cvename.c
gi?name=CVE-2006-4980
```

`glsa-check` ist in der Lage, aus einer GLSA die notwendigen Schritte für das Absichern des Systems abzuleiten und automatisch durchzuführen. In einem ersten Schritt sollte man sich die vorgeschlagenen Aktionen mit der `--pretend`-Option (bzw. `-p`) anzeigen lassen:

```
gentoo ~ # glsa-check -p $(glsa-check -t all)
This system is affected by the following GLSAs:
Checking GLSA 200611-06
The following updates will be performed for this GLSA:
     net-misc/openssh-4.4_p1-r6 (4.3_p2-r1)

Checking GLSA 200609-13
The following updates will be performed for this GLSA:
     app-arch/gzip-1.3.5-r9 (1.3.5-r8)

Checking GLSA 200609-17
The following updates will be performed for this GLSA:
     net-misc/openssh-4.3_p2-r5 (4.3_p2-r1)

Checking GLSA 200610-07
The following updates will be performed for this GLSA:
     dev-lang/python-2.4.3-r4 (2.4.3-r1)
```

Und schließlich werden die Korrekturen mit `--fix` (bzw. `-f`) durchgeführt.

```
gentoo ~ # glsa-check -f $(glsa-check -t all)
```

Im Wesentlichen geht es darum, automatisch `emerge` aufzurufen. So gerüstet, sollte es deutlich einfacher sein, das eigene System sicher zu halten.

11.1.3 eclean

Kommen wir zu den beiden Skripten `eclean` und `eread`. Das erste hilft dabei, das eigene System aufzuräumen.

Wie bereits unter 1.3.1 angesprochen, hat das Verzeichnis `/usr/portage/distfiles` (bzw. der Pfad in `DISTDIR` in der Datei `/etc/make.conf`) die

Tendenz, im Leben eines Gentoo-Systems deutlich anzuwachsen. Einige der dort abgelegten Quellarchive haben einen beachtlichen Umfang, und vor allem bei Desktop-Systemen können durch DISTDIR einige Gigabyte Speicherplatz verloren gehen.

Es gibt mehrere Varianten, diesen Speicherplatz zu sparen. Die sicherlich einfachste besteht darin, das Verzeichnis in regelmäßigen Abständen zu leeren. Die wohl radikalste – die Sie an dieser Stelle nicht durchführen sollten – ist folgende:

```
gentoo ~ # rm /usr/portage/distfiles/*
```

Damit reduziert man den Speicherverbrauch deutlich, zwingt Portage jedoch auch dazu, bei jedem emerge-Vorgang die Quellen erneut herunterzuladen. Wer die höhere Belastung des Netzwerks vermeiden möchte, verwendet eclean für einen Mittelweg.

eclean identifiziert im Standard-Modus all jene Quellarchive, zu denen es keinen passenden Ebuild mehr im Portage-Baum gibt, und löscht diese veralteten Pakete.

Um sich in einem ersten Schritt die Dateien anzeigen zu lassen, die eclean entfernen würde, lassen wir das Skript, wie bei den meisten Portage-Tools, mit der Option --pretend (bzw. -p) laufen und geben mit distfiles an, dass wir uns um die Dateien im DISTDIR kümmern möchten. Abhängig von der Zahl der zu überprüfenden Dateien kann dieser Vorgang eine Weile dauern und zeigt dann zuletzt die Menge Speicherplatz, die eingespart würde. Wir zeigen hier das Listing von einem anderen System, das schon etwas länger unter Gentoo arbeitet und ein paar alte Quellarchive angesammelt hat:

```
gentoo ~ # eclean -p distfiles
 * Building file list for distfiles cleaning...
 * Here are distfiles that would be deleted:
 [    0L B ] .keep
 [   3.9 K ] 01-add-2.6-devfs-and-sysfs-to-lirc_dev.patch
 [  63.4 K ] 20021129-cvs.diff.bz2
 [   2.6 K ] 3qe-1.0.tar.bz2
 [  37.4 M ] AdobeReader_enu-7.0.1-1.i386.rpm
...
 [ 180.6 K ] zina-0.12.10.tar.gz
 [ 363.0 K ] zlib-1.2.2.tar.bz2
 [  21.6 K ] zoo-2.10-gcc33-issues-fix.patch
 [ 510.2 K ] zvbi-0.2.4.tar.bz2
 * Total space that would be freed in distfiles directory: 2.0 G
```

Der zweite Lauf ohne die Option --pretend räumt das System auf. Die dabei entfernten Quellarchive sind in jedem Fall überflüssiger Ballast, da es keinen Ebuild mehr gibt, der diese Quellen je benutzen würde.

Wem der so gewonnene Speicherplatz nicht genügt, der kann `eclean` auch mit der Option `--destructive` (bzw. `-d`) aufrufen. In diesem Fall erhält `eclean` nur die Quellarchive, die zu aktuell installierten Paketen gehören. Damit ist Portage beim Downgrade eines Paketes gezwungen, das Quellpaket erneut herunterzuladen. Da dieses Ereignis nicht zu häufig eintritt, erhöht der Verlust dieser Quellarchive die Netzlast im Normalfall nur minimal.

Wie sich das Aufräumen automatisieren lässt, erläutern wir im Kontext von `cron` im Kapitel 16.6.4.

11.1.4 eread

Damit bleibt noch das kleine Werkzeug `eread` vorzustellen, das dem Auswerten von Portage-Log-Dateien dient. Dafür ist es allerdings notwendig (wie in Kapitel 6.3.4 beschrieben), `PORT_LOGDIR` zu definieren und unter `PORTAGE_ELOG_SYSTEM` den Wert `save` hinzuzufügen.

Die Installationsmeldungen eines jeden Paketes speichert `emerge` dann in `$PORT_LOGDIR/elog`, und wir können sie mit `eread` lesen bzw. verwalten.

```
gentoo ~ # grep "PORT.*LOG" /etc/make.conf
PORT_LOGDIR="/var/log/portage"
PORTAGE_ELOG_SYSTEM="save_summary save"
gentoo ~ # emerge app-portage/gentoolkit
...
gentoo ~ # eread

This is a list of portage log items. Choose a number to view that file o
r type q to quit.

1) summary.log
2) app-portage:gentoolkit-0.2.3:20080131-071004.log
Choice? 2
LOG: postinst

Another alternative to qpkg and equery are the q applets in
app-portage/portage-utils


WARN: postinst
The qpkg and etcat tools are deprecated in favor of equery and
are no longer installed in /usr/bin in this release.
They are still available in /usr/share/doc/gentoolkit-0.2.3/deprecated/
if you *really* want to use them.

app-portage:gentoolkit-0.2.3:20080131-071004.log lines 1-12/12 (END) q
Delete file? [y/N] y
Deleted app-portage:gentoolkit-0.2.3:20080131-071004.log
```

```
This is a list of portage log items. Choose a number to view that file o
r type q to quit.

1) summary.log
Choice? q
Quitting
```

Hier die Log-Information der Installation von app-portage/gentoolkit. Nach der Auswahl der Log-Datei mit [1] befinden wir uns im Viewer-Programm less und verlassen den Modus mit [Q] wieder. eread fragt uns dann, ob wir die Datei behalten oder löschen möchten. Da sie keine interessanten Informationen enthält, löschen wir sie mit [Y] und verlassen das Programm anschließend mit [Q].

11.2 Das Paket app-portage/portage-utils

Wir haben uns weiter oben ausführlich mit equery beschäftigt und festgestellt, dass dieses Skript bei manchen Operationen nicht eben schnell ist.

app-portage/portage-utils liefert einige kleine Werkzeuge, die in C geschriebene, beschleunigte Varianten mancher Funktionen von equery bereitstellen. Das Paket ist nicht dazu gedacht, equery zu ersetzen, sondern legt bei reduziertem Funktionsumfang Wert auf deutlich höhere Geschwindigkeit. Damit sind die Tools aus diesem Paket ideal für die Verwendung in eigenen Skripten.

qlist ist z. B. der Ersatz für equery files, liefert allerdings keine eigene Filterfunktion. Um zu erfahren, welche Werkzeuge das Paket installiert, kombinieren wir den qlist Aufruf mit dem grep-Befehl:

```
gentoo ~ # qlist app-portage/portage-utils | grep /usr/bin
/usr/bin/q
/usr/bin/qpkg
/usr/bin/qmerge
/usr/bin/qpy
/usr/bin/quse
/usr/bin/qdepends
/usr/bin/qlop
/usr/bin/qlist
/usr/bin/qfile
/usr/bin/qatom
/usr/bin/qcheck
/usr/bin/qtbz2
/usr/bin/qgrep
/usr/bin/qsearch
/usr/bin/qglsa
/usr/bin/qxpak
/usr/bin/qsize
/usr/bin/qcache
```

Nicht alle diese Werkzeuge sind für den täglichen Gebrauch gedacht, aber auf die wichtigsten wollen wir kurz eingehen.

`qfile` ist z. B. ein flinker Ersatz für `equery belongs`:

```
gentoo ~ # qfile /usr/bin/qfile
app-portage/portage-utils (/usr/bin/qfile)
```

`quse` führt den mit `equery hasuse -p foo` vergleichbaren Suchvorgang deutlich schneller durch, bietet dafür aber auch nur diese Funktionalität und beschränkt sich nicht auf die installierten Pakete.

```
gentoo ~ # quse ssl
use: Updating ebuild cache ...
use: Finished 22852 entries in 127.489846 seconds
app-admin/conserver/conserver-8.1.14.ebuild pam ssl tcpd debug
app-admin/gkrellm/gkrellm-2.2.10.ebuild gnutls lm_sensors nls ssl X
app-admin/gkrellm/gkrellm-2.2.5.ebuild X nls ssl
app-admin/gkrellm/gkrellm-2.2.9-r1.ebuild gnutls nls ssl X
...
www-servers/webfs/webfs-1.20.ebuild ssl
www-servers/webfs/webfs-1.21.ebuild ssl threads
x11-misc/qterm/qterm-0.4.0.ebuild arts esd ssl
x11-misc/x11vnc/x11vnc-0.8.2-r1.ebuild jpeg zlib threads ssl crypt v4l x
inerama
x11-misc/x11vnc/x11vnc-0.8.3.ebuild jpeg zlib threads ssl crypt v4l xine
rama
x11-misc/x11vnc/x11vnc-0.8.4.ebuild jpeg zlib threads ssl crypt v4l xine
rama
xfce-extra/xfce4-mailwatch/xfce4-mailwatch-1.0.1.ebuild ssl
```

`qsearch` erlaubt es, nach bestimmten Zeichenketten in den Paketnamen zu suchen, und entspricht damit `emerge --search`. Mit der Suche nach Paketen beschäftigen wir uns allerdings erst in Kapitel 13.1 ab Seite 304. Dort gehen wir auch noch auf `qgrep` ein.

Die Ausgabe von `qsearch` ist sehr reduziert:

```
gentoo ~ # qsearch automake
sys-devel/automake Used to generate Makefile.in from Makefile.am
sys-devel/automake-wrapper wrapper for automake to manage multiple autom
ake versions
```

`qsize` ist mit `equery size` vergleichbar, `qcheck` erwartungsgemäß mit `equery check`. `qdepends` dagegen informiert im Standardmodus nicht wie `equery depends` darüber, welche Pakete vom dem angegebenen Ebuild abhängen, sondern zeigt dessen Abhängigkeiten an:

```
gentoo ~ # qdepends dev-libs/openssl
dev-libs/openssl-0.9.8d: sys-apps/diffutils >=dev-lang/perl-5
```

Die zu `equery depends` analoge Funktionalität erhält man erst, indem man die Option `--query` (bzw. `-Q`) hinzufügt.

```
gentoo ~ # qdepends -Q dev-libs/openssl
dev-lang/python-2.4.3-r4
dev-db/mysql-5.0.26-r2
net-nds/openldap-2.3.30-r2
mail-mta/ssmtp-2.61-r2
net-www/apache-2.0.58-r2
net-misc/wget-1.10.2
net-misc/openssh-4.5_p1-r1
```

Zuletzt zu `qlop`: Dieses Programm analysiert die Log-Datei von `emerge` (`/var/log/emerge.log`).

Zum einen können wir uns mit der Option `--list` (bzw. `-l`) ansehen, welche Pakete wir installiert haben. Umgekehrt liefert uns `--unlist` (bzw. `-u`) die Pakete, die wir deinstalliert haben:

```
gentoo ~ # qlop -l
...
Wed Jan 30 13:51:04 2008 >>> app-portage/portage-utils-0.1.23
Wed Jan 30 13:51:35 2008 >>> app-arch/unzip-5.52-r1
Wed Jan 30 13:51:44 2008 >>> dev-java/java-config-wrapper-0.12-r1
Wed Jan 30 13:51:58 2008 >>> dev-java/java-config-2.0.31
Wed Jan 30 13:52:11 2008 >>> dev-java/java-config-1.3.7
Wed Jan 30 14:36:13 2008 >>> dev-libs/openssl-0.9.8d
Thu Jan 31 08:10:07 2008 >>> app-portage/gentoolkit-0.2.3
gentoo ~ # qlop -u
Tue Jan 29 20:25:59 2008 <<< dev-libs/openssl-0.9.8d
Wed Jan 30 08:33:21 2008 <<< dev-libs/openssl-0.9.7l
```

Wann wir den Portage-Baum synchronisiert haben, zeigt `qlop` mit der Option `--sync` (bzw. `-s`). Auch hier stammt das Beispiel von einem anderen System, da wir ja (noch) nicht synchronisiert haben:

```
gentoo ~ # qlop -s
...
Sun Jan 27 08:25:41 2008 >>> rsync://88.198.224.205/gentoo-portage
Mon Jan 28 11:17:22 2008 >>> rsync://88.198.224.205/gentoo-portage
Tue Jan 29 09:43:04 2008 >>> rsync://193.190.198.20/gentoo-portage
Wed Jan 30 09:53:21 2008 >>> rsync://147.32.127.222/gentoo-portage
```

Falls wir gerade dabei sind, ein Paket zu installieren, würde `qlop` mit der Option `--current` (bzw. `-c`) darüber Auskunft geben, um welches es sich handelt.

Sehr nützlich ist `qlop`, um vorab zu erfahren, wie lange `emerge` für das Aktualisieren eines Paketes vermutlich brauchen wird. Voraussetzung ist

zwar, dass man dieses Paket zuvor schon einmal kompiliert hat, aber wenn dies der Fall ist, liefert `qlop` mit der Option `--time` (bzw. `-t`) eine hilfreiche Zusammenfassung:

```
gentoo ~ # qlop -H -t dev-libs/openssl
openssl: 11 minutes, 30 seconds for 4 merges
```

Die Option `--human` (bzw. `-H`) listet die Zeitangabe in Stunden und Minuten statt in Sekunden und macht die Zusammenfassung damit verständlicher.

Wir haben hier `dev-libs/openssl` offensichtlich schon vier Mal auf dem System kompiliert. Wer die einzelnen Zeiten angezeigt bekommen möchte, greift zu der Option `--gauge` (bzw. `-g`):

```
gentoo ~ # qlop -H -g dev-libs/openssl
openssl: Tue Jan 29 20:16:51 2008: 9 minutes, 8 seconds
openssl: Wed Jan 30 08:18:00 2008: 15 minutes, 21 seconds
openssl: Wed Jan 30 10:09:51 2008: 8 minutes, 40 seconds
openssl: Wed Jan 30 14:23:20 2008: 12 minutes, 53 seconds
openssl: 4 times
```

So lässt sich schnell abschätzen, ob man das Update einer Software noch einschieben kann, bevor man den Rechner herunterfahren möchte.

Damit haben wir die wichtigsten Werkzeuge, die externe Pakete für das Paketmanagement unter Gentoo liefern, behandelt. Nun noch zu drei erwähnenswerten Skripten im zentralen Paket `sys-apps/portage`.

11.3 Das Paket sys-apps/portage

Eigentlich haben wir uns schon ausgiebig mit `sys-apps/portage` beschäftigt, liefert es doch `emerge`, unser primäres Tool für das Paketmanagement.

Es installiert zudem Programme wie `env-update` (Kapitel 9.1.2), `emerge-webrsync` (Kapitel 10.1.1), `ebuild` (Kapitel 15.1.2) und `etc-update` bzw. `dispatch-conf` (beide Kapitel 10.3). Drei weitere, `emaint`, `regenworld` und `quickpkg`, haben wir bisher noch nicht vorgestellt und wollen dies nun nachholen.

11.3.1 emaint

`emaint` erledigt Instandhaltungsaufgaben – derzeit beherrscht es allerdings noch nicht allzu viele davon. Um genau zu sein: nur eine einzige. Es kann die `world`-Datei (siehe Kapitel 10.2 auf Seite 217) überprüfen.

```
gentoo ~ # emaint --check world
Checking world for problems
Finished
```

Im Normalfall sollte das Ergebnis wie oben aussehen. Falls sich nicht existente Paketnamen in der Datei befinden, bemängelt `emaint` dies und kann das Problem mit der Option `--fix` beheben:

```
gentoo ~ # echo "broken
> " >> /var/lib/portage/world
gentoo ~ # emaint --check world
Checking world for problems

'broken' is not a valid atom


Finished
gentoo ~ # emaint --fix world
Attempting to fix world
Finished
gentoo ~ # emaint --check world
Checking world for problems
Finished
```

Wir haben hier als Beispiel der Datei `/var/lib/portage/world` einfach eine Zeile mit dem Inhalt `broken` hinzugefügt. Das entspricht keinem existierenden Paket und wird folglich von `emaint` moniert bzw. entfernt.

11.3.2 regenworld

Mit der `world`-Datei beschäftigt sich auch `regenworld`. Dieses Programm arbeitet sich durch die Log-Datei `/var/log/emerge.log` und identifiziert alle Pakete, die der Benutzer jemals manuell installiert hat. Fehlen Elemente in der `world`-Datei, kann `regenworld` diese hinzufügen.

Da `regenworld` vor dem Einsatz ein Backup empfiehlt, kopieren wir die Datei zunächst einmal:

```
gentoo ~ # regenworld --help
This script regenerates the portage world file by checking the portage
logfile for all actions that you've done in the past. It ignores any
arguments except --help. It is recommended that you make a backup of
your existing world file (var/lib/portage/world) before using this tool.
gentoo ~ # cp /var/lib/portage/world /var/lib/portage/world.backup
```

Anschließend klauen wir der `world`-Datei mit Hilfe von `grep` den `app-portage/gentoolkit`-Eintrag und lassen `regenworld` danach laufen:

```
gentoo ~ # cat /var/lib/portage/world.backup | grep -v "gentoolkit" > \
> /var/lib/portage/world
gentoo ~ # regenworld
add to world: app-portage/gentoolkit
```

`regenworld` identifiziert das entfernte Paket als fehlend und fügt es automatisch wieder zur `world`-Datei hinzu.

11.3.3 quickpkg

Dieses kleine Hilfsmittel kümmert sich darum, von einem installierten Paket einen Snapshot zu erstellen, den wir in Notfällen auch wieder dazu verwenden können, ein zerstörtes System wieder zu beleben.

So kann man z. B. `sys-libs/glibc` sichern:

```
gentoo ~ # quickpkg sys-libs/glibc
 * Building package for glibc-2.5 ...           [ ok ]

 * Packages now in /usr/portage/packages:
 * glibc-2.5: 11M
```

Das fertig gepackte Paket erstellt `quickpkg` in `/usr/portage/packages` (bzw. `PKGDIR`; siehe Kapitel 6.1 auf Seite 143). Dafür muss das Verzeichnis allerdings auch existieren.

```
gentoo ~ # ls /usr/portage/packages/sys-libs/
glibc-2.5.tbz2
```

Dieses Archiv enthält jetzt ein exaktes Abbild der Dateien des installierten Paketes und wir können es so nutzen, um unser System wieder herzustellen, wenn wir `sys-apps/glibc` beschädigt haben sollten.

Letzteres kann allerdings nur durch grobe Unachtsamkeit des Benutzers passieren, und wir wollen uns an dieser Stelle auch nicht mit den Details einer solchen Rettungsaktion befassen. Das Gentoo-Forum und auch das Gentoo-Wiki liefert für diese Fälle detaillierte Anleitungen.

11.4 Das Paket app-admin/eselect

Damit wechseln wir vom eigentlichen Paketmanagement zur Konfiguration des Systems und schließen das Kapitel mit einigen Bemerkungen zu `eselect`, einem noch relativ neuen, aber viel versprechenden Werkzeug.

Wir haben `eselect` auch schon zweimal erwähnt: Einmal bei der Behandlung der Init-Skripte in Kapitel 7.2 auf Seite 169 und bei der Profilwahl in Kapitel 5.2.1.

Wir können das Werkzeug mit dem Paket `app-admin/eselect` installieren (siehe Seite 129):

```
gentoo ~ # emerge -av app-admin/eselect

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-admin/eselect-1.0.7  USE="-bash-completion -doc" 0 k
B

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
```

`eselect` ist ein allgemeines Konfigurationstool und modular aufgebaut. Die Grundinstallation des Paketes `app-admin/eselect` bringt schon einige Standard-Module mit. Wir können sie unter Angabe der Aktion `list-modules` anzeigen:

```
gentoo ~ # eselect list-modules
Built-in modules:
  help                      Display a help message
  list-modules              Find and display available modules
  usage                     Display a usage message
  version                   Display version information

Extra modules:
  bashcomp                  Manage contributed bash-completion scripts
  binutils                  Manage installed versions of sys-devel/binut
ils
  env                       Manage environment variables set in /etc/env
.d/
  java-nsplugin             Manage the Java plugin for Netscape-like Bro
wsers
  java-vm                   Manage the Java system and user VM
  kernel                    Manage the /usr/src/linux symlink
  mailer                    Manage the mailwrapper profiles in /etc/mail
  profile                   Manage the /etc/make.profile symlink
  rc                        Manage /etc/init.d scripts in runlevels
```

Interessant sind hier nur die `Extra modules`; die oberen vier Module sind generelle Funktionen von `eselect`.

Die beiden Module `profile` und `rc` haben wir schon in Kapitel 5.2.1 bzw. 7.2 besprochen.

Bei jedem Modul erfahren wir etwas über die möglichen Option, indem wir `eselect` nur mit dem Modulnamen aufrufen, so z. B. für das `kernel`-Modul:

```
gentoo ~ # eselect kernel
Usage: eselect kernel <action> <options>

Standard actions:
  help                      Display help text
  usage                     Display usage information
  version                   Display version information

Extra actions:
  list                      List available kernel symlink targets
  set <target>              Set a new kernel symlink target
    target                    Target name or number (from 'list' action)
  show                      Show the current kernel symlink
```

Die meisten Module verrichten sehr einfache Aufgaben, die sich vor allem an Linux-Anfänger richten. Wer sich einigermaßen mit der Kommandozeile auskennt hat sicherlich kein Probleme damit, den Kernel-Symlink `/usr/src/linux` mit Hilfe von `ln` selber zu setzten, wenn er eine neue Version der Kernel-Quellen installiert hat.

`eselect` erlaubt es aber, Aktionen dieser Art in einem vereinheitlichten Format durchzuführen, und bietet sich damit auch als Backend für grafische Benutzeroberflächen zur Systemkonfiguration an.

11.4.1 Das Modul kernel

Das Kernel-Modul setzt den Symlink `/usr/src/linux` auf eine der installierten Kernel-Quellen.

Die Aktion `list` zeigt die verfügbaren Kernel-Versionen an:

```
gentoo ~ # eselect kernel list
Available kernel symlink targets:
  [1]   linux-2.6.19-gentoo-r5 *
```

Das Sternchen markiert die aktuell ausgewählte Version.

Über die Aktion `set` lässt sich die Kernel-Version ändern:

```
gentoo ~ # eselect kernel set linux-2.6.19-gentoo-r5
```

Hier ist das wenig sinnvoll, da ohnehin nur eine Kernel-Version zur Verfügung steht, aber das Konzept sollte klar sein.

11.4.2 Das Modul bashcomp

Eine der sehr nützlichen Eigenschaften der Shell `bash` ist das automatische Vervollständigen begonnener Befehle. Möchte man z. B. die Datei

`datei_mit_besonders_langem_dateinamen.txt` anzeigen und befindet sich sonst keine Datei, deren Name mit dem Buchstaben `d` beginnt, im selben Verzeichnis, genügt die Angabe `cat d` gefolgt von einem Anschlag der Tabulatortaste.

`bash` komplettiert dann automatisch zu `cat datei_mit_besonders_langem_dateinamen.txt`, und man hat sich Einiges an Tipparbeit gespart. Mit ein wenig Übung lohnt sich der Anschlag der Tabulatortaste dann auch schon, wenn nur noch zwei oder drei Buchstaben ausstehen.

Gibt es mehr als eine Möglichkeit der Vervollständigung (also z. B. eine zweite Datei mit dem Anfangsbuchstaben `d`), zeigt `bash` beim zweiten Anschlag der Tabulatortaste alle möglichen Komplettierungen an.

`bash` komplettiert in der Grundeinstellung nur Dateinamen, doch lässt sich die Komplettierung benutzerspezifisch anpassen und erweitern. Dafür installieren wir zunächst das Paket `app-shells/gentoo-bashcomp`:

```
gentoo ~ # emerge -av app-shells/gentoo-bashcomp

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-shells/bash-completion-20050121-r10  0 kB
[ebuild  N    ] app-shells/gentoo-bashcomp-20050516  0 kB

Total: 2 packages (2 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
 * Add the following to your ~/.bashrc to enable completion support.
 * NOTE: to avoid things like Gentoo bug #98627, you should set aliases
 * after sourcing /etc/profile.d/bash-completion.
 *
 * [[ -f /etc/profile.d/bash-completion ]] &&
 *     source /etc/profile.d/bash-completion
 *
 * Additional completion functions can be enabled by installing
 * app-admin/eselect and using the included bashcomp module.
...
```

In den älteren Versionen des Paketes ist es noch notwendig, das Paket manuell in die eigene Konfiguration einzubinden, indem wir die oben abgebildete Zeile in `~/.bashrc` übernehmen. Das ist mit den neueren Versionen auf einem aktualisierten System nicht mehr notwendig.

Hier editieren wir `~/.bashrc` also einmal kurz mit `nano` und lesen die Datei mit `source` ein, damit sich die neue Einstellung in unserer aktiven Shell bemerkbar macht. Dann kopieren wir noch das Standard-`bash`-Profil (`/etc/skel/.bash_profile`), so dass die Bash `~/.bashrc` beim nächsten Login automatisch einliest:

```
gentoo ~ # cat ~/.bashrc
[[ -f /etc/profile.d/bash-completion ]] && source /etc/profile.d/bash
-completion
gentoo ~ # source ~/.bashrc
gentoo ~ # cp /etc/skel/.bash_profile ~/.bash_profile
```

Damit ist es dann z. B. möglich, auch `emerge`-Befehle zeitsparend einzutippen. Möchten wir z. B. das Kernel-Paket `sys-kernel/gentoo-sources` installieren und sind uns bei der Bezeichnung nicht ganz sicher bzw. wissen nur noch, dass der Name mit `gentoo` begann, dann starten wir mit `emerge gentoo`, gefolgt von zwei Anschlägen der Tabulatortaste:

```
gentoo ~ # emerge gentoo[Tab][Tab]
gentoo                  gentoo-guide-xml-dtd    gentoo-sources
gentoo-artwork          gentoo-init             gentoo-syntax
gentoo-artwork-livecd   gentoolkit              gentoo-vdr-scripts
gentoo-bashcomp         gentoolkit-dev          gentoo-webroot-default
gentoo-bugger           gentoo-rsync-mirror     gentoo-xcursors
```

Nun zeigt Bash automatisch alle Pakete, die mit `gentoo` beginnen; wenn wir den Eintrag `gentoo-sources` sehen, fällt uns hoffentlich wieder ein, dass dies der korrekte Name des Kernel-Paketes war. In dem Fall komplettieren wir auf `gentoo-so` und betätigen die Tabulatortaste, damit `bash` den Rest des Namens hinzufügt.

Zurück zum `bashcomp`-Modul für `eselect`: Die Bash-Komplettierung ist ebenfalls modular, und wir können verschiedene Sets an Komplettierungen für verschiedene Programme aktivieren. Einen Überblick erhalten wir erst einmal mit der Aktion `list`. Die Option `--global` zeigt mit einem Sternchen an, welche Module global für alle Benutzer aktiviert sind:

```
gentoo ~ # eselect bashcomp list --global
Available completions:
  [1]   bitkeeper
  [2]   bittorrent
  [3]   cksfv
  [4]   clisp
  [5]   dsniff
  [6]   freeciv
  [7]   gcl
  [8]   gentoo *
  [9]   gkrellm
  [10]  gnatmake
  [11]  harbour
  [12]  isql
  [13]  larch
  [14]  lilypond
  [15]  lisp
  [16]  mailman
```

```
[17]  mcrypt
[18]  mtx
[19]  p4
[20]  povray
[21]  ri
[22]  sbcl
[23]  sitecopy
[24]  snownews
[25]  unace
[26]  unrar
```

Viele Module sind sehr spezifisch für bestimmte Programme. `gentoo` ist standardmäßig aktiviert und liefert eben z. B. die oben beschriebene Funktionalität für `emerge`.

Bei neueren `eselect`-Versionen wird `eselect` hier auch ein Modul gleichen Namens anzeigen. Wir können dieses dann aktivieren und uns die Verwendung von `eselect` etwas vereinfachen.

Aktivieren lässt sich ein Modul mit `enable` und dem Modulnamen. Ohne die Option `--global` aktivieren wir das Modul nur für den aufrufenden Benutzer. Mit der Option aktivieren wir es für alle Benutzer:

```
gentoo ~ # eselect bashcomp enable --global eselect
```

Hinweis für Systeme mit Netzwerkanbindung

Das `bashcomp`-Modul mit Namen `eselect` können Sie nur dann aktivieren, wenn Sie Ihr System bereits aktualisiert und `app-admin/eselect` in der neuesten Version installiert haben.

Beim nächsten Aufruf sollte `eselect ba` gefolgt von einem Anschlag der Tabulatortaste dann automatisch auf `eselect bashcomp` vervollständigen. Die Aktion `disable` deaktiviert bei Bedarf ein Modul wieder.

11.4.3 Das Modul env

Dieses Modul bietet nur eine mögliche Aktion: `update`, das die gleiche Funktion bietet wie `env-update` (siehe Seite 193).

11.4.4 Andere Module

Das Modul `binutils` erlaubt es unter den verschiedenen Versionen des Paketes `sys-devel/binutils` auszuwählen (sofern wirklich mehrere Versionen installiert sind).

Und schließlich kann man über das Modul `mailer` zwischen verschiedenen Mailprogrammen für die Kommandozeile wählen. Standardmäßig ist aber nur `mail-mta/ssmtp` installiert und eine Wahl erübrigt sich.

11.4.5 Weitere Module installieren

Wie gesagt ist `eselect` modular ausgelegt, und es gibt mittlerweile eine ganze Reihe an Ergänzungen für die verschiedenartigsten Konfigurationsaufgaben.

Eine Übersicht liefert `qsearch` (siehe Seite 304):

```
gentoo ~ # qsearch eselect
app-admin/eselect Modular -config replacement utility
app-admin/eselect-blas BLAS module for eselect
app-admin/eselect-cblas C-language BLAS module for eselect
app-admin/eselect-compiler Utility to configure the active toolchain com
piler
app-admin/eselect-esd Manages configuration of ESounD implementation or
PulseAudio wrapper
app-admin/eselect-gnat gnat module for eselect.
app-admin/eselect-lapack LAPACK module for eselect
app-admin/eselect-oodict Manages configuration of dictionaries for OpenO
ffice.Org.
app-admin/eselect-opengl Utility to change the OpenGL interface being us
ed
app-admin/eselect-oracle Utility to change the Oracle SQL*Plus Instantcl
ient being used
app-admin/eselect-timidity Manages configuration of TiMidity++ patchsets
app-admin/eselect-vi Manages the /usr/bin/vi symlink
net-wireless/waveselect Waveselect is wireless lan connection tool for L
inux using QT and wireless-tools.
```

Die gebräuchlichste Erweiterung für Desktop-Systeme liefert `app-admin/eselect-opengl`, mit dem man Treiber für die 3D-Beschleunigung auswählt.

Kapitel

12

Einen Webserver einrichten

Nun steht das notwendige Rüstzeug bereit, um unser frisches Gentoo-System zu administrieren, und es ist an der Zeit, Anwendungssoftware zu installieren und die Maschine ihrem eigentlichen Zweck zuzuführen. Die Möglichkeiten sind hier nahezu unbegrenzt, und jeder Nutzer wird wohl sein eigenes, auf seine speziellen Anforderungen angepasstes Set an Programmen zusammenstellen. Wie man die dazu notwendige Software bzw. die entsprechenden Pakete identifiziert, erklären wir im nächsten Kapitel.

Hier greifen wir zunächst – als ein mögliches Beispiel unter vielen – die Einrichtung eines Webservers heraus. Gentoo bietet eine recht fortschrittliche Webserver-Umgebung und liefert mit `webapp-config` ein Gentoo-spezifisches Werkzeug zur Installation von Web-Applikationen.

Wir wollen uns also an dieser Stelle ein wenig mit der Konfiguration des Apache-Servers auseinander setzen und dabei vor allem auf die spezielle Konfigurationsstruktur des Webservers unter Gentoo eingehen. Die Konfiguration von PHP beleuchten wir dann nur kurz, bevor wir zuletzt detailliert auf `webapp-config` eingehen.

12.1 Die Apache-Konfiguration

Die Apache-Konfiguration befindet sich unter `/etc/apache2`, zumindest wenn wir den Apache in der Version 2 installiert haben. Gentoo unterstützt auch die älteren 1.3.*-Varianten, aber wir wollen uns hier auf die neuere Version konzentrieren.

```
gentoo ~ # ls -la /etc/apache2/
insgesamt 80
drwxr-xr-x  5 root root  4096 29. Jan 14:00 .
drwxr-xr-x 45 root root  4096 31. Jan 08:52 ..
-rw-r--r--  1 root root  2068 29. Jan 16:03 apache2-builtin-mods
-rw-r--r--  1 root root 37741 29. Jan 16:03 httpd.conf
-rw-r--r--  1 root root 12958 29. Jan 16:03 magic
drwxr-xr-x  2 root root  4096 29. Jan 16:03 modules.d
drwxr-xr-x  2 root root  4096 29. Jan 14:00 ssl
drwxr-xr-x  2 root root  4096 29. Jan 14:00 vhosts.d
```

In dem Verzeichnis befindet sich die zentrale Apache-Konfigurationsdatei `httpd.conf`. Sie entspricht im Wesentlichen der Standardkonfiguration, wurde aber an einigen Stellen für Gentoo angepasst.

Eine Modifikation ist für die Struktur der Apache-Konfiguration unter Gentoo sehr wichtig: Alle Dateien, die auf `.conf` enden und entweder in `/etc/apache2/modules.d` oder in `/etc/apache2/vhosts.d` liegen, bezieht der Apache-Server in seine Hauptkonfiguration ein.

Diese beiden Verzeichnisse haben eine jeweils spezifische Funktion: So dient `/etc/apache2/modules.d` z. B. der erweiterten Konfiguration einzelner Module.

```
gentoo ~ # ls -la /etc/apache2/modules.d/
insgesamt 28
drwxr-xr-x 2 root root 4096 29. Jan 16:03 .
drwxr-xr-x 5 root root 4096 29. Jan 14:00 ..
-rw-r--r-- 1 root root 2980 29. Jan 16:03 40_mod_ssl.conf
-rw-r--r-- 1 root root 8151 29. Jan 16:03 41_mod_ssl.default-vhost.conf
-rw-r--r-- 1 root root  583 29. Jan 16:03 45_mod_dav.conf
-rw-r--r-- 1 root root  892 29. Jan 16:03 46_mod_ldap.conf
-rw-r--r-- 1 root root    0 29. Jan 16:03 .keep_net-www_apache-2
```

In der Standardkonfiguration finden sich hier nur Konfigurationsdateien für das SSL- (wenn wir Apache mit dem `ssl`-USE-Flag installiert haben) und das DAV-Modul.

Der Vorteil eines solchen Verzeichnisses liegt darin, dass weitere Apache-Pakete hier Konfigurationsdateien ablegen können, die der Apache-Server automatisch in die zentrale Konfiguration einbezieht, ohne dass der Nutzer über einen Editor eingreifen und sich selber mit der Grundkonfiguration auseinander setzen müsste.

Installieren wir z. B. ein zusätzliches, optionales Modul für den beschleunigten Ablauf von Perl-Web-Applikationen, `net-www/mod_perl`:

```
gentoo ~ # emerge -av www-apache/mod_perl

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] perl-core/CGI-3.25  230 kB
[ebuild  N    ] dev-perl/Compress-Raw-Zlib-2.001  0 kB
[ebuild  N    ] virtual/perl-Scalar-List-Utils-1.18  0 kB
[ebuild  N    ] dev-perl/Apache-Test-1.29  148 kB
[ebuild  N    ] app-admin/sudo-1.6.8_p12-r1  USE="ldap pam -offensive (-
selinux) -skey" 0 kB
[ebuild  N    ] dev-perl/IO-Compress-Base-2.001  0 kB
[ebuild  N    ] virtual/perl-CGI-3.25  0 kB
[ebuild  N    ] dev-perl/IO-Compress-Zlib-2.001  0 kB
[ebuild  N    ] dev-perl/Compress-Zlib-2.001  0 kB
[ebuild  N    ] www-apache/mod_perl-2.0.3-r1  3,628 kB

Total: 10 packages (10 new), Size of downloads: 4,006 kB

Would you like to merge these packages? [Yes/No] Yes
```

Hinweis für Systeme mit Netzwerkanbindung

Sie benötigen eine funktionieren Netzwerkverbindung, um das Paket hier zu installieren.

und schauen wir uns nun mit `qlist` und `grep` eine Auswahl der installierten Dateien an:

```
gentoo ~ # qlist www-apache/mod_perl | grep modules
/etc/apache2/modules.d/apache2-mod_perl-startup.pl
/etc/apache2/modules.d/75_mod_perl.conf
/usr/lib/apache2/modules/mod_perl.so
```

Das eigentliche Modul findet sich unter `/usr/lib/apache2/modules`, die Dokumentation unter `/usr/share/doc` und die Modul-eigene Konfiguration wandert in `/etc/apache2/modules.d`. Die dort abgelegte Konfigurationsdatei ist dafür zuständig, das Modul selbst zu laden und die Grundkonfiguration festzulegen.

Das zweite Verzeichnis `/etc/apache2/vhosts.d` dient der Definition virtueller Hosts. Prinzipiell sollte hier eine Datei pro virtuellen Host angelegt und entsprechend benannt werden. Letztlich ist es aber jedem selbst überlassen, ob er dieser Struktur folgt.

Abgesehen vom Verzeichnis /etc/apache2 findet sich noch eine weitere wichtige Konfigurationsdatei für den Apache unter /etc/conf.d/apache2. Die zentrale Variable ist hier APACHE2_OPTS. Dort lassen sich komfortabel bestimmte Eigenschaften der Apache-Konfiguration aktivieren bzw. deaktivieren. Dafür sind innerhalb der Konfigurationsdaten Blöcke mit einem <IfDefine OPTION> umgeben. Solch einen Block können wir dann über den Eintrag -D OPTION in der Variable APACHE2_OPTS aktivieren bzw. deaktivieren.

In httpd.conf gibt es beispielsweise die Option USERDIR, die das Mapping der Verzeichnisse /home/*username*/public_html auf http://www.example.com/~*username* aktiviert.

Eine andere Option, INFO, aktiviert dagegen die zwei Pfade http://www.example.com/server-status und http://www.example.com/server-info, die für einen Überblick interner Webserverdaten dienen.

Als Basiskonfiguration können wir zusätzlich zu diesen beiden Optionen die Konfiguration für einen Standard-Host sowie die SSL-Unterstützung aktivieren:

```
APACHE2_OPTS="-D DEFAULT_VHOST -D INFO -D USERDIR -D SSL \
-D SSL_DEFAULT_VHOST"
```

Der hier aktivierte Standard-Host (DEFAULT_VHOST) liefert die Webdateien aus dem Verzeichnis /var/www/localhost/htdocs aus. Die Option SSL aktiviert das SSL-Modul, während die Einstellung SSL_DEFAULT_VHOST einen virtuellen SSL-Host auf Port 443 aktiviert und ebenfalls die Dateien aus dem Verzeichnis /var/www/localhost/htdocs anbietet.

Zuletzt ist der Webserver über das init-Skript zu starten:

```
gentoo ~ # /etc/init.d/apache2 start
 * Starting apache2 ...                                [ ok ]
```

12.2 PHP

Da die meisten Web-Applikationen PHP als Skript-Sprache verwenden, wollen wir das Paket dev-lang/php hier installieren und auch als Modul für den Apache-Webserver hinzufügen.

PHP besitzt eine Vielzahl unterschiedlicher Module, die den Satz an Basisfunktionen der Skriptsprache erweitern. Unter Gentoo können wir diese Erweiterungen über USE-Flags aktivieren bzw. deaktivieren. Viele Web-Applikationen brauchen spezifische Erweiterungen von PHP, und so sollte man bei der Installation einen guten Satz häufig benötigter USE-Flags

aktivieren, um die Wahrscheinlichkeit zu reduzieren, zu einem späteren Zeitpunkt PHP erneut kompilieren zu *müssen*, nur weil eine neue Web-Applikation bestimmte PHP-Funktionen vermisst.

Eine weitere Schwierigkeit war bei PHP der Wechsel von Version 4 auf Version 5. Da sich die Entwickler von PHP dazu entschlossen haben, keine vollständige Kompatibilität zwischen den beiden Sprachversionen zu gewährleisten, bot Gentoo die Möglichkeit, beide Versionen gleichzeitig zu installieren und parallel zu nutzen.

Da die alte PHP-Version mittlerweile leider keine sicherheitsrelevanten Korrekturen mehr bekommt, wurde PHP 4 aber vollständig aus dem Gentoo-Angebot entfernt und PHP 5 ist nun die einzige unterstützte Version.

Hier ein Vorschlag für USE-Flags, die man in `/etc/portage/package.use` für PHP festlegen kann:

```
gentoo ~ # echo "dev-lang/php apache2 cgi force-cgi-redirect cli  \
> berkdb bzip2 calendar crypt ctype curl doc exif ftp gd gdbm iconv imap\
> json ldap mhash mysql ncurses nls pcre pdo readline session simplexml \
> soap spell sqlite ssl tokenizer truetype unicode xml xsl zlib" >> \
> /etc/portage/package.use
```

Für die eigene Installation sollte man sich vor der Festlegung der USE-Flags zumindest kurz über die verfügbaren Optionen informieren, da sich diese auch bei kleineren Versionssprüngen ändern können.

In den oben angegebenen USE-Flags verstecken sich drei Einstellungen, die sich nicht auf spezielle Erweiterungen der Sprache PHP beziehen, sondern Einfluss auf die Installationsweise von PHP nehmen. Und zwar aktiviert die Option `apache2` das entsprechende Apache-Modul, während `cgi` das Modul für die Behandlung von PHP-Programmen als CGI-Skript aktiviert. Das USE-Flag `cli` erstellt gleichzeitig den PHP-Command-Line-Parser.

Nachdem wir die USE-Flags festgelegt haben, können wir nun das PHP-Modul installieren:

```
gentoo ~ # emerge -av dev-lang/php

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] dev-libs/libxml2-2.6.27  USE="ipv6 python readline -debu
g -doc -test" 0 kB
[ebuild  N    ] media-libs/t1lib-5.0.2  USE="-X -doc" 0 kB
[ebuild  N    ] dev-db/sqlite-2.8.16-r4  USE="nls -doc -tcl" 959 kB
[ebuild  N    ] dev-libs/libgpg-error-1.0-r1  USE="nls" 0 kB
[ebuild  N    ] dev-libs/libmcrypt-2.5.7  512 kB
[ebuild  N    ] app-crypt/mhash-0.9.2  834 kB
[ebuild  N    ] app-text/aspell-0.50.5-r4  USE="gpm" 0 kB
[ebuild  N    ] net-libs/c-client-2004a-r1  USE="pam ssl" 2,173 kB
```

```
[ebuild  N    ] app-admin/php-toolkit-1.0-r2  0 kB
[ebuild  N    ] dev-db/sqlite-3.3.5-r1  USE="-debug -doc -nothreadsafe -
tcl" 0 kB
[ebuild  N    ] net-misc/curl-7.15.1-r1  USE="ipv6 ldap ssl -ares -gnutl
s -idn -kerberos -krb4 -test" 0 kB
[ebuild  N    ] dev-libs/libgcrypt-1.2.2-r1  USE="nls" 0 kB
[ebuild  N    ] dev-libs/libxslt-1.1.17  USE="crypt python -debug" 0 kB
[ebuild  N    ] dev-lang/php-5.2.1-r3  USE="apache2 berkdb bzip2 calenda
r cgi cli crypt ctype curl doc exif force-cgi-redirect ftp gd gdbm iconv
 imap ipv6 json ldap mhash mysql ncurses nls pcre pdo readline reflectio
n session simplexml soap spell spl sqlite ssl tokenizer truetype unicode
 xml xsl zlib -adabas -apache -bcmath -birdstep -cdb -cjk -concurrentmod
php -curlwrappers -db2 -dbase -dbmaker -debug -discard-path -empress -em
press-bcs -esoob -fastbuild -fdftk -filter -firebird -flatfile -frontbas
e -gd-external -gmp -hash -inifile -interbase -iodbc -java-external -ker
beros -ldap-sasl -libedit -mcve -msql -mssql -mysqli -oci8 -oci8-instant
-client -odbc -pcntl -pdo-external -pic -posix -postgres -qdbm -recode -
sapdb -sharedext -sharedmem -snmp -sockets -solid -suhosin -sybase -syba
se-ct -sysvipc -threads -tidy -wddx -xmlreader -xmlrpc -xmlwriter -xpm -
yaz -zip -zip-external" 7,019 kB
[ebuild  N    ] app-doc/php-docs-20050822  2,678 kB

Total: 15 packages (15 new), Size of downloads: 14,172 kB

Would you like to merge these packages? [Yes/No] Yes
```

Hinweis für Systeme mit Netzwerkanbindung

Sie benötigen eine funktionieren Netzwerkverbindung, um das Paket hier zu installieren.

Zur Modul-Konfiguration installiert Portage dann die Datei `/etc/apache2/modules.d/70_mod_php.conf`. Die darin enthaltene Standardkonfiguration müssen wir nicht bearbeiten, aber wir sollten das PHP-Modul für den Apache-Server mit `-D PHP` in `/etc/conf.d/apache2` aktivieren:

```
APACHE2_OPTS="-D DEFAULT_VHOST -D INFO -D USERDIR -D SSL \
-D SSL_DEFAULT_VHOST -D PHP"
```

Wenn der Server bereits lief, müssen wir ihn nach Veränderungen der Konfiguration in `/etc/conf.d/apache` neu starten:

```
gentoo ~ # /etc/init.d/apache2 restart
 * Stopping apache2 ...                                [ ok ]
 * Starting apache2 ...                                [ ok ]
```

Nun wird der Apache automatisch alle Dateien, die auf `.php` enden, auch wirklich als PHP-Skripte ausführen.

12.3 Dateien ausliefern

Sobald der Webserver läuft, können wir Applikationen in den Verzeichnissen platzieren, die Apache der Außenwelt präsentiert.

Machen wir es uns zunächst einmal einfach und testen den virtuellen Host, den wir oben mit `-D DEFAULT_VHOST` aktiviert haben. Wie bereits erwähnt, liegen die auszuliefernden Dateien für diesen virtuellen Host in `/var/www/localhost/htdocs` (d. h. `DocumentRoot` wurde auf den entsprechenden Pfad festgelegt). `net-www/apache` hat das Verzeichnis schon rudimentär für uns vorbereitet.

Unter `/var/www/localhost` finden sich die Verzeichnisse `htdocs`, `cgi-bin` und `icons`. Innerhalb der `icons` installiert `net-www/apache` einige Apache-Standardgrafiken, `cgi-bin` erhält einige Test-CGI-Skripte, während `htdocs` eine simple `index.html`-Datei enthält, die anzeigt, dass hier ein Apache-Server sein Werk verrichtet.

Wenn wir den Server weiter oben ohne Probleme starten konnten, sollte er sich jetzt mit einem Browser ansprechen lassen und darüber informieren, dass der Apache-Server läuft, jedoch noch kein Inhalt vorhanden ist. Wir testen dies kurz mit dem textbasierten Browser `www-client/links`.

`links` ist zwar auf der LiveDVD verfügbar, aber in unserem neu aufgesetzten System ist das Paket noch nicht installiert. Das holen wir an dieser Stelle nach. Wer als Kommandozeilen-Browser `lynx` bevorzugt, kann hier natürlich auch das Paket `www-client/lynx` installieren.

```
gentoo ~ # emerge -av www-client/links

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] www-client/links-2.1_pre26  USE="gpm ssl unicode -X -dir
ectfb -fbcon -javascript -jpeg -livecd -png -sdl -svga -tiff" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
gentoo ~ # links http://localhost/
```

Sie sollten jetzt von `links` eine Nachricht gezeigt bekommen, dass die Installation des Apache-Servers erfolgreich war.

Damit können wir in `/var/www/localhost/htdocs` manuell weitere Dateien oder ganze PHP-Applikationen installieren. Gentoo bietet hier ein weiteres Werkzeug, um die Installation von Web-Applikationen zu vereinfachen. Mit diesem wollen wir uns im Folgenden beschäftigen.

12.4 webapp-config

Für die Installation von Web-Anwendungen ist oft eine Vielzahl verschiedener Schritte notwendig. Zentral ist natürlich das Ablegen der Programmdateien im Web-Verzeichnis (in unserem Fall also in `/var/www/localhost/htdocs`). Doch bei vielen Web-Applikation kommt das Einrichten einer Datenbank, das Konfigurieren des Servers und auch das Konfigurieren der Applikation selbst hinzu.

Für die sehr verbreitete Kombination aus *L*inux, *A*pache-Server, *M*ySQL-Server und *P*HP-Applikationen (*LAMP*) gibt es leider kein Standard-Werkzeug, das genau diese Einrichtung erleichtern oder automatisieren würde. Darum nehmen viele Nutzer die Installation von PHP-Web-Applikationen weiterhin manuell vor. Dies ist zwar für den erfahrenen Benutzer durchaus machbar, folgt aber eigentlich nicht der Philosophie einer Distribution, und schon gar nicht der von Gentoo, denn kaum ein Nutzer käme analog dazu auf die Idee, Gentoo ohne den Paketmanager Portage zu installieren, obwohl dies möglich ist. Für Web-Applikationen versucht das Werkzeug `webapp-config`, die Einrichtung zu vereinfachen und zu automatisieren.

Eines vorweg: `webapp-config` ist mit Sicherheit kein Universalwerkzeug, das die Einrichtung von Web-Applikation plötzlich zum Kinderspiel machen würde. So ist `webapp-config` z. B. nicht in der Lage, Datenbanken automatisch zu konfigurieren oder die Konfiguration des Webservers bei Bedarf zu modifizieren. Denn die vielen verschiedenen Applikationen unterscheiden sich schon von vornherein stark in der Verbindung mit MySQL oder in den Anforderungen an die Apache-Konfiguration. Zusätzlich werden diese oft noch individuell vom Anwender angepasst. Automatisierung ist hier also nahezu ein Ding der Unmöglichkeit.

`webapp-config` ist aber sehr gut zu gebrauchen für das Dateimanagement bei der Installation, das Entfernen oder auch das Upgrade von Web-Applikationen. Es übernimmt im Wesentlichen die Aufgaben von Portage im besonderen Fall von Web-Anwendungen.

Portage würde es dem Nutzer lediglich ermöglichen, eine Web-Applikation einmal zu installieren, z. B. in `/var/www/localhost/htdocs`. Würden wir auf die Idee kommen, einen zweiten virtuellen Host unter `/var/www/example.com/htdocs` anzulegen und dort die gleiche PHP-Anwendung installieren wollen, wären wir wieder zur Handarbeit gezwungen.

An dieser Stelle setzt `webapp-config` an und installiert eine Web-Applikation als *Master*-Kopie, um diese in die verschiedenen virtuellen Hosts des Systems zu kopieren. Es wird dabei festhalten, welche Dateien wo installiert wurden, so dass bei einer Deinstallation ein sauberes System zurückbleibt. Außerdem übernimmt das Tool bei einem Software-Upgrade genau die gleichen Funktionen wie Portage und schützt z. B. die Konfigurationsdateien vor unbeabsichtigtem Überschreiben.

Soviel zur Einleitung. Installieren wir app-admin/webapp-config erst einmal. Wir wählen hier die instabile Version, da wir andernfalls später auf einen Fehler des Programms treffen würden:

```
gentoo ~ # flagedit app-admin/webapp-config -- +~x86
gentoo ~ # emerge -av app-admin/webapp-config

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-admin/webapp-config-1.50.16  95 kB

Total: 1 package (1 new), Size of downloads: 95 kB

Would you like to merge these packages? [Yes/No] Yes
```

Hinweis für Systeme mit Netzwerkanbindung

Sie benötigen eine funktionieren Netzwerkverbindung, um das Paket hier zu installieren.

Die meisten Applikationen hängen übrigens von webapp-config ab, so dass die Applikation im Normalfall auch mit der ersten installierten Web-Anwendung installiert wird.

12.4.1 webapp-config konfigurieren

In der Regel benötigt die Konfigurationsdatei (/etc/vhosts/webapp-config) keine Modifikationen. Als Basis-Hostname (vhost_hostname) wählt das Programm z. B. erst einmal localhost aus und setzt auf dieser Basis das Standard-Installationsverzeichnis vhost_root dann auf /var/www/${vhost_hostname}. Ergänzen wir das Verzeichnis für Applikationen, die wir nicht in einem SSL-Host installieren wollen (vhost_htdocs_insecure mit dem Wert htdocs), gelangen wir zu unserem Standard-Webserver-Verzeichnis /var/www/localhost/htdocs.

Da sich die wichtigsten Parameter auch über die webapp-config-Kommandozeile festlegen lassen, müssen wir an diesen Einstellungen im Normalfall nichts variieren.

12.4.2 Web-Applikationen installieren

Installieren wir nun unsere erste Web-Applikation. Die meisten Web-Anwendungen finden sich in der Kategorie www-apps, und wir wollen an die-

ser Stelle das Paket www-apps/zina zur Verwaltung von mp3-Dateien installieren. Es ist in Sachen Funktionalität keine herausragende Web-Applikation, aber der Ebuild bietet alle wichtigen Eigenschaften, die eine Web-Applikation unter Gentoo haben kann. Außerdem ist das Paket recht klein und besitzt keine aufwendige Konfiguration.

Wir installieren erst einmal das Paket und stellen dabei sicher, dass wir das vhosts-USE-Flag deaktivieren (wir werden dieses ab Seite 284 detailliert erklären). Das Paket ist als instabil markiert, deshalb fügen wir es noch mit flagedit zu /etc/portage/package.keywords hinzu:

```
gentoo ~ # flagedit www-apps/zina -- +~x86
gentoo ~ # USE="-vhosts" emerge -av www-apps/zina

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] www-apps/zina-1.0_rc2  USE="-vhosts" 194 kB

Total: 1 package (1 new), Size of downloads: 194 kB

Would you like to merge these packages? [Yes/No] Yes
```

Hinweis für Systeme mit Netzwerkanbindung

Sie benötigen eine funktionieren Netzwerkverbindung, um das Paket hier zu installieren.

Gegen Ende der Installation meldet sich der emerge-Prozess mit:

```
 * vhosts USE flag not set - auto-installing using webapp-config
 * Running //usr/sbin/webapp-config -I -h localhost -u root -d /zina zin
a 1.0_rc2
```

Was passiert an dieser Stelle? Um eine Antwort zu finden, können wir mit equery files zina anzeigen, welche Dateien das Paket überhaupt installiert hat. Dabei stellen wir fest, dass nahezu alle Dateien irgendwo unter /usr/share/webapps abgelegt wurden. Dies scheint vielleicht ein eher ungewöhnlicher Ort für die Installation einer Web-Applikation zu sein. Man würde eher ein Verzeichnis unter /var/www oder /srv/www erwarten.

Doch schauen wir unter /var/www nach, so stellen wir fest, dass unter /var/www/localhost/htdocs/zina offenbar eine komplette Zina-Installation liegt:

```
gentoo ~ # la /var/www/localhost/htdocs/zina
insgesamt 251
```

```
drwxr-xr-x 4 root root     4096 12. Jan 15:04 .
drwxr-xr-x 5 root root     4096 12. Jan 15:04 ..
-rwxr-xr-x 2 root root      805 12. Jan 15:04 .htaccess
-rwxr-xr-x 2 root root   170993 12. Jan 15:04 index.php
drwxr-xr-x 3 root root     4096 12. Jan 15:04 music
-rw------- 1 root root      304 12. Jan 15:04 .webapp
-rw------- 1 root root    27968 12. Jan 15:04 .webapp-zina-1.0_rc2
drwxr-xr-x 6 root root     4096 12. Jan 15:04 _zina
-rwxr-xr-x 2 root root    35398 12. Jan 15:04 zina_db.php
-rw-rw-r-- 1 root apache      0 12. Jan 15:04 zina.ini.php
```

Das ist beruhigend, denn schließlich ist `/var/www` der Standardpfad für Dateien, die der Webserver ausliefern soll.

Trotzdem erscheint es merkwürdig, dass `emerge` die Applikation zweimal installiert hat. Das ist in diesem Fall aber auch nicht ganz korrekt. Der eigentliche Installationsort liegt, wie auch über `equery files` nachprüfbar ist, in `/usr/share/webapps/zina`. In `/var/www/localhost/htdocs/zina` befindet sich eine sogenannte *virtuelle Installation*. Schaut man genauer auf das oben gezeigte Datei-Listing, so sieht man z. B an der 2 hinter den Zugriffsrechten der `index.php` Datei, dass der Inhalt der Datei mit zwei Pfaden im System verknüpft ist.

Mit anderen Worten sind `/var/www/localhost/htdocs/zina/index.php` und `/usr/share/webapps/zina/1.0_rc2/htdocs/index.php` dieselbe Datei, aber durch eine harte Verknüpfung zweimal im Dateisystem abgebildet.

12.4.3 Der Ort der Installation

Wozu dieser Mechanismus? Diese Art, die Pakete zu installieren, soll vor allem Nutzer unterstützen, die ein und dieselbe Web-Applikation mehrfach über den Apache-Server anbieten wollen. Dies ist meist der Fall, wenn man mehrere virtuelle Hosts über den Apache betreibt und mehrere Domänen gleichzeitig unterstützt.

Nehmen wir einmal an, wir wollen die Domänen `test1.example.com` und `test2.example.com` mit demselben Apache-Server bedienen und `www-apps/zina` in beiden Domänen anbieten. Dann sollten wir zuerst einmal die beiden virtuellen Hosts innerhalb der Apache-Konfiguration definieren. Dafür benutzen wir das auf Seite 272 erwähnte Verzeichnis `/etc/apache2/vhosts.d`.

Legen wir im Stil von `/etc/apache2/vhosts.d/00_default_vhost.conf` die Datei `50_test1.example.com` mit folgendem Inhalt an:

```
<VirtualHost *:80>
    DocumentRoot "/var/www/test1.example.com/htdocs"
    <Directory "/var/www/test1.example.com/htdocs">
```

```
        # Indexes: Erlaubt Apache ein Inhaltsverzeichnis
        #          eines Verzeichnisses anzuzeigen, wenn
        #          keine konkrete Datei ausgewählt wurde.
        # FollowSymLinks: Erlaubt Apache auch symbolischen
        #                 Links zu folgen
        Options Indexes FollowSymLinks

        # Verbietet das Überschreiben zentraler
        # Konfigurationsdirektiven über .htaccess Dateien
        AllowOverride None
        # Zugriff auf die Dateien in diesem Verzeichnis erlauben
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

Nach dem gleichen Schema legt man die zweite Datei `51_test2.example.com` an und modifiziert die `test1.example.com`-Einträge zu `test2.example.com`.

Um unsere Maschine vorübergehend zu überzeugen, lokal generierte Anfragen auf `test1.example.com` und `test2.example.com` zu beantworten, fügen wir in der Datei `/etc/hosts` in der Zeile, die mit `127.0.0.1` beginnt, die beiden Test-Adressen hinzu:

```
gentoo ~ # cat /etc/hosts
127.0.0.1  localhost localhost.localdomain test1.example.com test2.examp
le.com
```

Danach starten wir den Apache-Server neu, um die oben hinzugefügten Konfigurationen einzulesen:

```
gentoo ~ # /etc/init.d/apache2 restart
```

Als `DocumentRoot` haben wir dabei in den oben angelegten Dateien also `/var/www/`*`DOMAIN-NAME`*`/htdocs` festgelegt. Das entspricht dem Standard-Gentoo-Schema.

Mit `webapp-config` können wir Zina nun in diesen zwei virtuellen Hosts installieren. Dazu müssen wir Zina *nicht* noch einmal mit `emerge` installieren! Folgender Befehl reicht für die Installation aus:

```
gentoo ~ # webapp-config -I -h test1.example.com -d /  zina 1.0_rc2
*
* You may be installing into the website's root directory.
* Is this what you meant to do?
*
*   Creating required directories
*   Linking in required files
```

```
*     This can take several minutes for larger apps
*   Files and directories installed

=========================================================================
POST-INSTALL INSTRUCTIONS
=========================================================================

-----------------
  CONFIGURATION
-----------------

There is not much to do for Zina to work "out of the box" - just
browse to

http://test1.example.com//

and follow the simple instructions.

MAKE SURE TO CHANGE YOUR ADMIN PASSWORD!

-----------------
  SECURITY NOTE
-----------------

You are highly encouraged to restrict access to your Zina (unless you
actually WANT to have the world see your audio file collections). Zina
has no internal user-based authentication - only a single
administrative user and password.

-----------------
      NOTES
-----------------

This is the standalone Zina installation. For any embedded use, please
see "http://www.pancake.org/zina/" (Section "Installation").

You might also want to grab some sort of album cover fetching
application, like "AlbumArt" (in Portage as media-sound/albumart).

=========================================================================

* Install completed - success
gentoo ~ # webapp-config -I -h test2.example.com -d /  zina 1.0_rc2
...
```

Die zweite Installation erfolgt dann genauso, nur ersetzen wir `test1` wieder durch `test2`.

In beiden Fällen instruieren wir `webapp-config` mit der Option `--install` (bzw. `-I`), die Applikation zu installieren. Dabei gibt dann `--host` (bzw. `-h`) den virtuellen Host (bzw. eigentlich den Ordner unter `/var/www`) an, in dem wir die Applikation installieren möchten.

webapp-config hat während der Installation nun die Verzeichnisse /var/www/test1.example.com/htdocs und /var/www/test2.example.com/htdocs angelegt und in ihnen jeweils Zina installiert. Schauen wir noch einmal die in der Installation enthaltene index.php-Datei an, sehen wir, dass diese nun viermal verlinkt ist, einmal unter /usr/share/webapps, dann unter /var/www je einmal in localhost, test1.example.com und test2.example.com:

```
gentoo ~ # ls -la /var/www/test1.example.com/htdocs/index.php
-rwxr-xr-x 4 root root 170993 12. Jan 15:04 /var/www/test1.example.com/ht
docs/index.php
```

Mit diesem Vorgehen sparen wir also Speicherplatz, da webapp-config die Dateien nur von einem zentralen Ort aus verlinkt.

```
gentoo ~ # du -c -s -h /usr/share/webapps/zina/
1,5M /usr/share/webapps/zina/
1,5M insgesamt
```

Im Falle von zina halten sich die Einsparungen noch in Grenzen, aber bei Applikationen wie Gallery2 spart man pro Installation schon ca. 36 MB.

12.4.4 vhosts USE-Flag

Damit haben wir den wohl verwirrendsten Aspekt einer webapp-config-Installation geklärt. Darüber hinaus fehlt uns aber noch ein tieferer Einblick in die Benutzung des Skripts. Wenden wir uns zunächst dem oben verwendeten vhosts-USE-Flag zu. Wie der Name suggeriert, handelt es sich hier um *virtuelle Hosts*. Dieses USE-Flag wird dazu verwendet anzuzeigen, ob emerge webapp-config schon bei der eigentlichen Installation einer Web-Applikation aufrufen soll oder nicht.

Schauen wir uns einfach noch einmal die vorangegangene Installation an und aktivieren diesmal das vhosts-USE-Flag:

```
gentoo ~ # USE="vhosts" emerge -av www-apps/zina

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] www-apps/zina-1.0_rc2  USE="vhosts*" 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
```

Diesmal erfolgt die Installation ohne den Aufruf von `webapp-config`. Stattdessen informiert uns `emerge`, dass die `vhosts`-Option aktiviert wurde, die Web-Applikation *nicht* automatisch installiert wird und wir die Applikation mit der angegebenen Kommandozeile in einen unserer virtuellen Hosts installieren können:

```
*
* The 'vhosts' USE flag is switched ON
* This means that Portage will not automatically run webapp-config to
* complete the installation.
*
* To install zina-1.0_rc2 into a virtual host, run the following command:
*
*     webapp-config -I -h <host> -d zina zina 1.0_rc2
*
* For more details, see the webapp-config(8) man page
```

Wer also nur eine Domäne mit seinem Apache-Server bedient, sollte das `vhosts`-USE-Flag nicht aktivieren. Dann installiert `emerge` die Web-Applikationen sofort unter `/var/www/localhost`, und diese sind damit direkt verfügbar. Wer mehr Kontrolle möchte aktiviert das entsprechende USE-Flag und kann somit frei entscheiden, an welchen Orten er die Web-Applikationen installieren möchte.

12.4.5 webapp-config anwenden

Bisher haben wir uns nur die beiden Optionen `--install` und `--host` angesehen.

Für den Installationsort ist darüber hinaus die Angabe des Zielverzeichnisses mit `-d` (bzw. `--dir`) wichtig. Über die Kombination des Host-Namens mit dem Zielverzeichnis bildet `webapp-config` den Zielpfad für die Installation.

Die Basis dieses Pfades bildet die Angabe für `vhost_root` in der Konfigurationsdatei unter `/etc/vhosts/webapp-config`. In der Standardeinstellung hat dieser Parameter den Wert `/var/www/${vhost_hostnames}` und enthält den gewählten virtuellen Hostnamen (`vhost_hostnames`), wie wir ihn auf der Kommandozeile mit der Option `-h` angegeben haben. Davon ausgehend, lautet der volle Pfad dann `${vhost_root}/${vhost_htdocs_insecure}/${installdir}`. `vhost_htdocs_insecure` findet sich ebenfalls in `/etc/vhosts/webapp-config` wieder und ist standardmäßig auf `htdocs` festgesetzt. Das `installdir` bezieht seinen Wert aus der `-d`-Option auf der Kommandozeile.

Mit diesem Schema unterstützt `webapp-config` jedoch nur das Standard-System der Installation von Web-Applikationen. Seltenere Varianten, wie

die Installation von Web-Applikationen in Benutzerverzeichnisse (beispielsweise `~/public_html`), unterstützt `webapp-config` derzeit nicht.

Wer zwischen einem normalen und einem SSL-gesicherten Host unterscheiden möchte, kann den Installationspfad zusätzlich über die Option `--secure` beeinflussen. Aktivieren wir diese Option, wird der Parameter `${vhost_htdocs_insecure}` im oben angegebenen Pfad durch `${vhost_htdocs_secure}` ersetzt. Der Standardwert ist hier im Normalfall `htdocs-secure`.

Den SSL-gesicherten virtuellen Host, der Dateien aus `htdocs-secure` anstatt `htdocs` liefert, muss man allerdings selbst in der Apache-Konfiguration erstellen.

Für die Installation einer Web-Anwendung sind zwingend zwei Argumente notwendig: der Name des Paketes und die Versionsnummer. Wir haben das weiter oben schon gesehen, als wir `zina` mit der Angabe `zina 1.0_rc2` installiert haben. Beide Argumente finden sich im Normalfall am Ende der Befehlszeile.

Damit haben wir erst einmal die wichtigsten Optionen von `webapp-config` behandelt. Uns fehlen aber noch die Befehle, um Applikationen wieder zu entfernen oder zu aktualisieren. Installieren wir einmal eine ältere Version des gleichen Tools:

```
gentoo ~ # USE="vhosts" emerge -av =www-apps/zina-0.12.12

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  NS   ] www-apps/zina-0.12.12  USE=''vhosts'' 193 kB

Total: 1 package (1 in new slot), Size of downloads: 193 kB

Would you like to merge these packages? [Yes/No] Yes
```

Diese Version installieren wir nun in einem zweiten Verzeichnis (`zina-old`) innerhalb des ersten Test-Hosts:

```
gentoo ~ # webapp-config -I -h test1.example.com \
> --pretend -d zina-old zina 0.12.12
...
gentoo ~ # webapp-config -I -h test1.example.com  \
> -d zina-old zina 0.12.12
...
```

Das erste Kommando enthält die Option `--pretend`, mit der wir die Installation testen, ohne tatsächlich Veränderungen am System vorzunehmen. `webapp-config` liefert dann etwas ausführlicheren Output und testet, ob alle Aktionen erfolgreich ausgeführt werden können. In diesem Beispiel

sollten keine Probleme auftreten. Wir können dann die `--pretend` Option entfernen und die Installation tatsächlich durchführen.

Würden wir die Installation jetzt verwenden und konfigurieren, dann würde die Web-Applikation einige Einstellungen in `/var/www/test1.example.com/htdocs/zina-old/zina.ini.php` schreiben. Wir wollen das hier der Einfachheit halber über die Kommandozeile erledigen und nehmen an, wir hätten den Benutzernamen des Administrators verändert und das Passwort neu gesetzt:

```
gentoo ~ # echo '<?php
> $adm_pwd = "{md5}jtM+aM68CE1F38JDDGnhOT5302s=";
> $adm_name = "wrobel";
> ?>' >> /var/www/test1.example.com/htdocs/zina-old/zina.ini.php
```

Damit simulieren wir eine konfigurierte Applikation, die wir im Folgenden aktualisieren wollen. Hierfür verwenden wir die Option `-U` (bzw. `--update`) anstelle des schon bekannten `-I`. Der Rest der Kommandozeile für `webapp-config` bleibt unberührt, nur die Versionsnummer müssen wir noch aktualisieren:

```
gentoo ~ # webapp-config -U -h test1.example.com -d zina-old zina 1.0_rc2
* Upgrading zina-0.12.12 to zina-1.0_rc2
*   Installed by root on 2007-01-16 15:40:27
*   Config files owned by 0:0
!time zina.ini.php
--- /var/www/test1.example.com/htdocs/zina-old
* Remove whatever is listed above by hand
*   Creating required directories
*   Linking in required files
*     This can take several minutes for larger apps
^o^ hiding /zina.ini.php
* zina-1.0_rc2 does not install any files from /usr/share/webapps/zina/1
.0_rc2/hostroot/; skipping
*   Files and directories installed
* One or more files have been config protected
* To complete your install, you need to run the following command(s):
*
* CONFIG_PROTECT="/var/www/test1.example.com/htdocs/zina-old//" etc-upda
te
*
* Install completed - success
```

Auch hier kann wer möchte den Lauf einmal vorher mit `--pretend` überprüfen und `webapp-config` erst dann wirklich zur Tat schreiten lassen.

Die Ausgabe zeigt, dass die von uns modifizierte Konfigurationsdatei von `webapp-config` geschützt wird. Das Programm fordert uns auf, die Datei mit den üblichen Methoden über `etc-update` zu aktualisieren:

```
gentoo ~ # CONFIG_PROTECT="/var/www/test1.example.com/htdocs/zina-old//"\
> etc-update
Scanning Configuration files...
The following is the list of files which need updating, each
configuration file is followed by a list of possible replacement
files.
1) /var/www/test1.example.com/htdocs/zina-old///zina.ini.php (1)
Please select a file to edit by entering the corresponding number.
              (don't use -3, -5, -7 or -9 if you're unsure what to do)
              (-1 to exit) (-3 to auto merge all remaining files)
                           (-5 to auto-merge AND not use 'mv -i')
                           (-7 to discard all updates)
                           (-9 to discard all updates AND not use 'rm -i
'): 1

Showing differences between /var/www/test1.example.com/htdocs/zina-old//
/zina.ini.php and /var/www/test1.example.com/htdocs/z
ina-old///._cfg0000_zina.ini.php
--- /var/www/test1.example.com/htdocs/zina-old///zina.ini.php   2008-02-
01 09:46:33.000000000 +0100
+++ /var/www/test1.example.com/htdocs/zina-old///._cfg0000_zina.ini.php
2008-02-01 09:46:41.000000000 +0100
@@ -1,4 +0,0 @@
-<?php
-$adm_pwd = "{md5}jtM+aM68CE1F38JDDGnhOT5302s=";
-$adm_name = "wrobel";
-?>
lines 1-8/8 (END) q

File: /var/www/test1.example.com/htdocs/zina-old///._cfg0000_zina.ini.ph
p
1) Replace original with update
2) Delete update, keeping original as is
3) Interactively merge original with update
4) Show differences again
Please select from the menu above (-1 to ignore this update): 1
Replacing /var/www/test1.example.com/htdocs/zina-old///zina.ini.php with
 /var/www/test1.example.com/htdocs/zina-old///._cfg0000_zina.ini.php
mv: ?/var/www/test1.example.com/htdocs/zina-old///zina.ini.php? überschr
eiben? y

Exiting: Nothing left to do; exiting. :)
```

Der Pfad zur Installation ist anzugeben, da dieser Ort normalerweise nicht in `CONFIG_PROTECT` (siehe Kapitel 10.3 ab Seite 219) enthalten ist.

`webapp-config` schützt ausschließlich Dateien, die explizit als Konfigurationsdateien markiert wurden. Zu dem entsprechenden Mechanismus kommen wir später ab Seite 292. An dieser Stelle sei nur erwähnt, dass `webapp-config` alle anderen veränderten Dateien bei einem Update überschreibt. Das Verhalten des Tools entspricht damit dem von `emerge` selbst.

Da wir unsere neu installierte Applikation nicht wirklich nutzen wollen, können wir auch auf das Update der Konfigurationsdateien verzichten und die Applikation gleich wieder deinstallieren. Dies geschieht über die Option `-C` (bzw. `--clean`).

```
gentoo ~ # webapp-config -C -h test1.example.com -d zina-old
* Removing zina-1.0_rc2 from /var/www/test1.example.com/htdocs/zina-old
*   Installed by root on 2007-01-16 17:00:21
*   Config files owned by 0:0
--- /var/www/test1.example.com/htdocs/zina-old
* Remove whatever is listed above by hand
```

Die Zeile mit dem Installationspfad direkt über der Meldung `Remove whatever is listed above by hand` erscheint nur, wenn man die Konfigurationsdateien nicht aktualisiert hat. Dann befinden sich im Installationspfad noch versteckte Dateien, die `webapp-config` davon abhalten, das Verzeichnis zu entfernen, da es nicht leer ist und `webapp-config` weiß, dass die Dateien nicht zum `zina`-Paket gehören. Wieder entspricht dies dem Verhalten von `emerge` bei der Deinstallation eines Paketes. Man kann hier aber gefahrlos die verbliebenen Reste der Testinstallation beseitigen:

```
gentoo ~ # rm -rf /var/www/test1.example.com/htdocs/zina-old
```

Damit haben wir die Hauptaktionen von `webapp-config` (Installation, Upgrade, Deinstallation) abgedeckt. Das Tool bietet darüber hinaus noch ein paar Hilfsfunktionen.

So kann man sich z. B. über `--list-installs` (bzw. `--li`) anzeigen lassen, welche Applikationen wo installiert sind:

```
gentoo ~ # webapp-config --li --verbose
* Installs for zina-1.0_rc2
*   /var/www/localhost/htdocs/zina
*   /var/www/test1.example.com/htdocs
*   /var/www/test2.example.com/htdocs
```

Wir sehen also, derzeit haben wir nur `zina` installiert – das allerdings an drei verschiedenen Orten – deren Pfade uns angegeben werden. Die zusätzliche Option `--verbose` (bzw. `-V`) liefert nicht nur die Installationspfade zurück, sondern zeigt darüber hinaus auch den Namen und die Version der installierten Web-Applikation an. Wollen wir nur die Installationspfade für `zina` wissen, hängen wir wie gewohnt den Namen der Applikation und die Version als Argumente an:

```
gentoo ~ # webapp-config --li zina 1.0_rc2
/var/www/localhost/htdocs/zina
/var/www/test1.example.com/htdocs
/var/www/test2.example.com/htdocs
```

Mit der Option `--list-unused-installs` (bzw. `--lui`) erhält man die Liste unbenutzter Web-Applikationen, also solcher Installationen, die nur unter `/usr/share/webapps` liegen und die wir in keinen der virtuellen Hosts unter `/var/www` installiert haben:

```
gentoo ~ # webapp-config --lui
zina-0.12.12
```

12.4.6 webapp-cleaner

Wir sehen hier, dass wir die veraltete `zina`-Version offensichtlich nicht mehr verwenden. Derzeit sind also zwei verschiedene `zina`-Versionen installiert:

```
gentoo ~ # ls /usr/share/webapps/zina/
0.12.12  1.0_rc2
```

Web-Applikationen sind offensichtlich Pakete, die `emerge` in *Slots* (siehe Kapitel 10.4.1) installiert und bei denen ein Update nicht zum Ersatz der älteren Version führt, denn es sollte ja möglich sein, auf verschiedenen virtuellen Hosts unterschiedliche Versionen der gleichen Web-Applikation zu betreiben. Ein Systemadministrator kann dann erst einmal die Software selbst unter `/usr/share/webapps` aktualisieren und die eigentlich installierten Web-Anwendungen Schritt für Schritt auf den neuesten Stand bringen. Veraltete Versionen sollten wir aber aus dem System entfernen, sobald wir sie nicht mehr benötigen. Dabei hilft das Werkzeug `webapp-cleaner`.

```
gentoo ~ # webapp-cleaner -p -C zina
 * Unused versions of zina detected.
 * To clean, run the following command:
 * emerge -Cav =zina-0.12.12
```

Die Option `--clean-unused` (bzw. `-C`) veranlasst `webapp-cleaner`, alle Versionen zu entfernen, zu denen es keine korrespondierenden virtuellen Installationen mehr gibt. Alternativ kann man auch mit `--prune` (bzw. `-P`) alle Versionen bis auf die neueste löschen. `--pretend` (bzw. `-p`) führt dazu, dass `webapp-cleaner` die Aktion nicht wirklich durchführt, sondern nur den notwendigen `emerge`-Befehl für die Aufräumaktion ausgibt und es dem Benutzer überlässt, ihn auszuführen.

An dieser Stelle können wir die alte `zina`-Installation aber durchaus entfernen:

```
gentoo ~ # webapp-cleaner -C zina
 * Unused versions of zina detected.
 * Running emerge -Cav =zina-0.12.12
```

```
>>> These are the packages that would be unmerged:

 www-apps/zina
    selected: 0.12.12
   protected: none
     omitted: 1.0_rc2

>>> 'Selected' packages are slated for removal.
>>> 'Protected' and 'omitted' packages will not be removed.

Would you like to unmerge these packages? [Yes/No] yes
```

`webapp-cleaner` geht ohne die `--pretend`-Option direkt in die Deinstallation mit `emerge` über.

12.4.7 webapp-config und Zugriffsrechte

Bisher haben wir eigentlich nur gesehen, dass `webapp-config` eine Web-Anwendung von einem zentralen Ort unter `/usr/share/webapps` in die virtuellen Hosts unter `/var/www` kopiert und Möglichkeiten bietet, diese Installationen zu verwalten. Das mag zwar recht nützlich sein, rechtfertigt aber nicht unbedingt ein eigenes Werkzeug.

Allerdings leistet `webapp-config` einiges, was nicht offensichtlich ist. Vergleichen wir dazu die Originalinstallation unter `/usr/share/webapps/zina/1.0_rc2/htdocs` mit der davon abgeleiteten unter `/var/www/localhost/htdocs/zina`:

```
gentoo ~ # ls -la /usr/share/webapps/zina/1.0_rc2/htdocs/
insgesamt 221
drwxr-xr-x 4 root root   4096 12. Jan 15:04 .
drwxr-xr-x 3 root root   4096 14. Jan 01:42 ..
-rwxr-xr-x 1 root root    805 14. Jan 01:42 .htaccess
-rwxr-xr-x 1 root root 170993 14. Jan 01:42 index.php
drwxr-xr-x 3 root root   4096 12. Jan 15:04 music
drwxr-xr-x 6 root root   4096 12. Jan 15:04 _zina
-rwxr-xr-x 1 root root  35398 14. Jan 01:42 zina_db.php
-rw-r--r-- 1 root root      0 14. Jan 01:42 zina.ini.php
gentoo ~ # ls -la /var/www/localhost/htdocs/zina/
insgesamt 251
drwxr-xr-x 4 root root     4096 12. Jan 15:04 .
drwxr-xr-x 5 root root     4096 12. Jan 15:04 ..
-rwxr-xr-x 3 root root      805 12. Jan 15:04 .htaccess
-rwxr-xr-x 3 root root   170993 12. Jan 15:04 index.php
drwxr-xr-x 3 root root     4096 12. Jan 15:04 music
-rw------- 1 root root      304 12. Jan 15:04 .webapp
-rw------- 1 root root    27968 12. Jan 15:04 .webapp-zina-1.0_rc2
drwxr-xr-x 6 root root     4096 12. Jan 15:04 _zina
-rwxr-xr-x 3 root root    35398 12. Jan 15:04 zina_db.php
-rw-rw-r-- 1 root apache      0 12. Jan 15:04 zina.ini.php
```

Abgesehen davon, dass `webapp-config` das Verzeichnis mit der versteckten Datei `.webapp` als installierte Web-Applikation markiert und einige Daten in der ebenfalls versteckten Datei `.webapp-zina-1.0_rc2` ablegt, fällt auf, dass der Eigentümer und die Zugriffsrechte der Konfigurationsdatei `zina.ini.php` verändert wurde. Die Datei ist für Mitglieder der Gruppe `apache` und damit vor allem unserem Webserver schreibbar. Das liegt daran, dass `webapp-config` die Datei `zina.ini.php` gesondert behandelt, wobei es für diese Spezialbehandlung zwei Konzepte gibt: `config-owned` und `server-owned`.

config-owned

Konfigurationsdateien sind bei Web-Applikation intern als `config-owned` markiert. Diese Markierung erfüllt zwei Zwecke:

- Die Dateien schützt `webapp-config` vergleichbar dem `CONFIG_PROTECT`-Mechanismus von `emerge` (siehe Kapitel 10.3).
- Die Dateien gehören dem Benutzer, für den die Applikation installiert wurde.

Den ersten Mechanismus haben wir im Zusammenhang mit der `zina`-Aktualisierung von Version `0.12.12` auf `1.0_rc2` kennen gelernt (siehe Seite 287). Dabei hat `webapp-config` die Konfigurationsdatei `zina.ini.php` nicht überschrieben und uns die Möglichkeit gegeben, sie mit `etc-update` zu bearbeiten.

Für den zweiten Mechanismus müssen wir nochmals eine Testinstallation vornehmen:

```
gentoo ~ # useradd -m wrobel
gentoo ~ # webapp-config -I -h test1.example.com -u wrobel \
-d zina-user zina 1.0_rc2
...
```

Die Option `--user` (bzw. `-u`) gibt den Benutzer an, für den wir die Installation durchführen. Schauen wir uns die virtuelle Installation an:

```
gentoo ~ # ls -la /var/www/test1.example.com/htdocs/zina-user
insgesamt 251
drwxr-xr-x 4 root   root     4096 17. Jan 08:23 .
drwxr-xr-x 6 root   root     4096 17. Jan 08:23 ..
-rwxr-xr-x 2 root   root      805 14. Jan 01:42 .htaccess
-rwxr-xr-x 2 root   root   170993 14. Jan 01:42 index.php
drwxr-xr-x 3 root   root     4096 17. Jan 08:23 music
-rw------- 1 root   root      314 17. Jan 08:23 .webapp
-rw------- 1 root   root    27675 17. Jan 08:23 .webapp-zina-1.0_rc2
```

```
drwxr-xr-x 6 root   root     4096 17. Jan 08:23 _zina
-rwxr-xr-x 2 root   root    35398 14. Jan 01:42 zina_db.php
-rw-rw-r-- 1 wrobel apache      0 17. Jan 08:23 zina.ini.php
```

Die Konfigurationsdatei `zina.ini.php` gehört nun dem angegebenen Nutzer und kann damit von ihm editiert werden. Alle anderen Dateien gehören aber weiterhin `root` und sind somit unzugänglich für den Nutzer. Der Benutzer darf also die für ihn installierte Applikation konfigurieren, nicht aber den eigentlichen Code modifizieren.

Mit der Option `--group` (bzw. `-g`) lässt sich die Gruppe der Konfigurationsdateien modifizieren. Im Falle von `zina.ini.php` bringt dies wenig, da die Datei gleichzeitig `server-owned` ist.

server-owned

Abgesehen von Konfigurationsdateien, die ein Benutzer editieren können sollte, gibt es noch Dateien oder auch Verzeichnisse, in denen der Webserver Schreibrechte haben sollte.

`zina` hat z. B. ein Cache-Verzeichnis, in dem die Applikation häufiger benötigte Daten ablegt, um wiederholte Prozesse zu beschleunigen. Schauen wir uns dieses Verzeichnis wieder im Vergleich zwischen der Originalinstallation unter `/usr/share/webapps/zina/1.0_rc2/htdocs` und der virtuellen Installation unter `/var/www/localhost/htdocs/zina` an:

```
gentoo ~ # ls -la /usr/share/webapps/zina/1.0_rc2/htdocs/_zina
insgesamt 28
drwxr-xr-x 6 root root 4096 12. Jan 15:04 .
drwxr-xr-x 4 root root 4096 12. Jan 15:04 ..
drwxr-xr-x 2 root root 4096 14. Jan 01:42 cache
drwxr-xr-x 2 root root 4096 12. Jan 15:04 lang
drwxr-xr-x 2 root root 4096 12. Jan 15:04 lang-cfg
-rwxr-xr-x 2 root root 1015 14. Jan 01:42 podcast.xml
drwxr-xr-x 5 root root 4096 12. Jan 15:04 themes
-rwxr-xr-x 2 root root 2583 14. Jan 01:42 zina.js
gentoo ~ # ls -la /var/www/localhost/htdocs/zina/_zina
insgesamt 28
drwxr-xr-x 6 root   root   4096 12. Jan 15:04 .
drwxr-xr-x 4 root   root   4096 12. Jan 15:04 ..
drwxr-xr-x 4 apache apache 4096 12. Jan 15:04 cache
drwxr-xr-x 2 root   root   4096 12. Jan 15:04 lang
drwxr-xr-x 2 root   root   4096 12. Jan 15:04 lang-cfg
-rwxr-xr-x 3 root   root   1015 12. Jan 15:04 podcast.xml
drwxr-xr-x 5 root   root   4096 12. Jan 15:04 themes
-rwxr-xr-x 3 root   root   2583 12. Jan 15:04 zina.js
```

Das entsprechende `cache`-Verzeichnis gehört in der virtuellen Installation dem Nutzer `apache`.

`webapp-config` kennt aber nicht nur den Apache. Die Option `--list-servers` (bzw. `--ls`) listet die unterstützten Server:

```
gentoo ~ # webapp-config --ls
apache
aolserver
lighttpd
cherokee
```

Über `--server` (bzw. `-s`) können wir für die Installation den passenden Servertyp auswählen. Alternativ verändert man die Standard-Einstellung `vhost_server` in `/etc/vhosts/webapp-config` auf einen anderen Wert als `apache`.

Abgesehen davon, dass `webapp-config` den korrekten Server-Benutzer auswählt, überprüft das Tool, ob das entsprechende Paket überhaupt installiert ist. Fehlt es, liefert `webapp-config` die notwendigen Anweisungen:

```
gentoo ~ # webapp-config -I -s lighttpd -h test1.example.com -u wrobel \
-d zina-server zina 1.0_rc2
* Fatal error: Your configuration file sets the server type "Lighttpd"
* Fatal error: but the corresponding package does not seem to be install
ed!
* Fatal error: Please "emerge www-servers/lighttpd" or correct your sett
ings.
* Fatal error(s) - aborting
```

config-owned + sever-owned = config-server-owned

Schließlich gibt es Dateien, die sowohl Konfigurationsdatei sind, also dem Nutzer gehören, zugleich aber dem Server Schreibrechte bieten sollen. Das ist vor allem dann sinnvoll, wenn die Web-Applikation eine Konfiguration über das Web ermöglicht, was z. B. bei `zina` der Fall ist. In diesem Fall muss der Web-Server in der Lage sein, die über die Webseite gewählten Einstellungen in der Konfigurationsdatei zu verewigen.

Dateien, die gleichzeitig als `config-owned` und `server-owned` markiert sind, haben den Nutzer als Eigentümer und werden der Server-Gruppe zugeordnet. Sowohl Eigentümer als auch Gruppenmitglieder erhalten dann Lese- und Schreibberechtigung auf die Datei.

```
gentoo ~ # ls -la /var/www/test1.example.com/htdocs/zina-user/zina.ini.php
-rw-rw-r-- 1 wrobel apache 0 17. Jan 08:23 /var/www/test1.example.com/htd
ocs/zina-user/zina.ini.php
```

12.4.8 Linktyp

Im Standard-Modus erzeugt `webapp-config`, wie weiter oben beschrieben, harte Verknüpfungen zwischen den Dateien. Diese Möglichkeit besteht unter einem Linux-Dateisystem aber nur, wenn die verknüpften Dateien im selben Dateisystem liegen.

Als Beispiel ein System, bei dem `/usr` und `/var` in verschiedenen Partitionen liegen:

```
gentoo ~ # mount
...
/dev/hda7 on /var type reiserfs (rw)
...
/dev/hda10 on /usr type reiserfs (rw)
...
```

Schauen wir uns auf diesem System die `zina`-Installation an, so sehen wir, dass die Dateien nicht verlinkt sind (Ziffer 1 hinter einfachen Dateien):

```
gentoo ~ # ls -la /var/www/localhost/htdocs/zina/
insgesamt 245
drwxr-xr-x  4 root root      288 12. Jan 14:23 .
drwxr-xr-x 13 root root      488 12. Jan 14:48 ..
-rwxr-xr-x  1 root root      805 12. Jan 14:20 .htaccess
-rwxr-xr-x  1 root root   170993 12. Jan 14:20 index.php
drwxr-xr-x  3 root root      144 12. Jan 14:20 music
-rw-------  1 root root      304 12. Jan 14:20 .webapp
-rw-------  1 root root    27968 12. Jan 14:20 .webapp-zina-1.0_rc2
drwxr-xr-x  6 root root      200 12. Jan 14:20 _zina
-rwxr-xr-x  1 root root    35398 12. Jan 14:20 zina_db.php
-rw-rw-r--  1 root apache    515 12. Jan 14:24 zina.ini.php
```

Sobald `webapp-config` merkt, dass es Dateien nicht ordnungsgemäß hart verlinken kann, wird als Ausweichsmodus einfach kopiert. Damit wird natürlich die eigentliche Idee, Speicherplatz zu sparen, zunichte gemacht, und der Nutzen von `webapp-config` besteht allein im Setzen der korrekten Zugriffsrechte.

Es besteht aber auch die Möglichkeit, von harten auf weiche Verknüpfungen (Soft Links oder Symlinks) zu wechseln. Diese sind auch zwischen Dateien auf unterschiedlichen Partitionen möglich. Dazu dient die Option `--soft` bei der Installation:

```
gentoo ~ # webapp-config -I --soft -d zina-soft zina 1.0_rc2
gentoo ~ # ls -la /var/www/localhost/htdocs/zina-soft/
insgesamt 45
drwxr-xr-x  4 root root     288 18. Jan 10:50 .
drwxr-xr-x 15 root root     544 18. Jan 10:50 ..
```

```
lrwxrwxrwx  1 root root      48 18. Jan 10:50 .htaccess -> /usr/share/we
bapps/zina/1.0_rc2/htdocs/.htaccess
lrwxrwxrwx  1 root root      48 18. Jan 10:50 index.php -> /usr/share/we
bapps/zina/1.0_rc2/htdocs/index.php
drwxr-xr-x  3 root root     144 18. Jan 10:50 music
-rw-------  1 root root     308 18. Jan 10:50 .webapp
-rw-------  1 root root   39834 18. Jan 10:50 .webapp-zina-1.0_rc2
drwxr-xr-x  6 root root     200 18. Jan 10:50 _zina
lrwxrwxrwx  1 root root      50 18. Jan 10:50 zina_db.php -> /usr/share/
webapps/zina/1.0_rc2/htdocs/zina_db.php
-rw-rw-r--  1 root apache     0 18. Jan 10:50 zina.ini.php
```

Bleibt die Frage, warum diese Option nicht Standardmodus von `webapp-config` ist. Das lässt sich bereits sehr schön an der gerade installierten `zina`-Version demonstrieren: Sie funktioniert nämlich nicht. Leider sind die wenigsten Web-Applikationen darauf ausgerichtet, mit symbolischen Verknüpfungen umzugehen. Zwar zeigt das eben installierte `zina` noch den gewohnten Text, aber es scheint vergessen zu haben, wo sich die Bilder und Style-Sheets befinden. Aus diesem Grunde ist die `--soft`-Option nur mit Vorsicht zu genießen.

12.4.9 Nachteile

Web-Applikationen auf Basis von Python oder Ruby haben bei der Installation Hilfswerkzeuge wie `webapp-config` deutlich weniger nötig, denn es gleicht eigentlich nur strukturelle Schwierigkeiten der Sprache PHP bzw. eines LAMP-Servers aus. Da eine Vielzahl interessanter Applikationen aber nun einmal in PHP geschrieben ist, ist es aus Distributionssicht sinnvoll, ein Werkzeug wie `webapp-config` anzubieten.

Einer der großen Nachteile von `webapp-config` ist aber dessen Komplexität. Viele Benutzer, die eine manuelle Installation von Web-Applikationen gewöhnt sind, fühlen sich durch die „doppelte Installation“ irritiert und empfinden das Skript meist als zusätzlichen Aufwand.

Erst nach einiger Zeit tritt dann häufig ein Gewöhnungseffekt auf und vor allem für die Betreiber mehrerer Domains wird das Tool dann schnell unverzichtbar. Allerdings ist das Werkzeug so nur für die Nutzer von Interesse, die virtuelle Hosts verwenden. Wer einen einzelnen Webserver mit einer einzelnen Domain betreibt zieht keinen Nutzen aus dem Tool.

Kapitel

13

Software finden und installieren

Im vorangegangenen Kapitel ging es exemplarisch um den Einsatz eines Gentoo-Systems als Webserver, doch mit Tausenden Software-Paketen, die uns zur Verfügung stehen, sind selbstverständlich auch ganz andere Szenarien umsetzbar. Wir können hier nur die grundlegenden Verfahren beschreiben, mit denen Sie die zu den gewünschten Funktionalitäten passenden Pakete identifizieren und in das System integrieren.

Im Falle von `apache` oder `mysql` mag dies intuitiv sein, da der Paketname mit dem Namen der Software korrespondiert. Wie aber findet man Pakete, wenn man allenfalls (grobe) Vorstellungen von den gewünschten Funktionen hat?

Sicher können Suchmaschinen im Internet weiterhelfen; wir wollen uns hier jedoch mit den Bordmitteln, d. h. mit der Kommandozeile unseres Systems behelfen und auf die Suche nach geeigneten Paketen machen.

13.1 emerge --search

Nehmen wir an, der mit der Standardinstallation verfügbare Editor `nano` entspricht nicht unseren Vorstellungen und wir arbeiten lieber mit `vi` oder `emacs`. Beide stecken in weit verbreiteten Paketen, so dass wir einfach versuchen, `emerge` die uns bekannten Namen mit auf den Weg zu geben.

Bei `emacs` führt das tatsächlich zum Erfolg:

```
gentoo ~ # emerge -av emacs

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-editors/emacs-21.4-r4  USE="nls -X -Xaw3d -leim -les
stif -motif -nosendmail" 19,925 kB

Total size of downloads: 19,925 kB

Would you like to merge these packages? [Yes/No]
```

Wir haben ausnahmsweise nicht die Kategorie (`app-editors`) angegeben, da wir den Paketnamen ja nur „raten“. Im Falle von `vi` scheitert der Versuch allerdings vollkommen:

```
gentoo ~ # emerge -av vi

These are the packages that would be merged, in order:

Calculating dependencies
emerge: there are no ebuilds to satisfy "vi".
```

Aus den Angaben von `emerge` im Emacs-Beispiel wissen wir, dass der Editor zur Kategorie `app-editors` gehört. Die Vermutung liegt also nahe, dass sich `vi` ebenfalls in dieser Kategorie befindet. Folglich könnte man die Verzeichnisse dieser Kategorie im Portage-Baum auflisten und die Ausgabe dadurch einschränken, dass nur solche Pakete erscheinen, die mit dem Buchstaben `v` beginnen:

```
gentoo ~ # ls /usr/portage/app-editors | grep "^v"
vile
vim
vim-core
```

Möglicherweise führt `emerge vim` (für *vi improved*) zum Ziel, was aber bedeutet dann der Eintrag `vim-core`? Der Paketname allein, wie `ls` ihn uns liefert, ist also nicht aussagekräftig genug.

Versuchen wir es mit der Suchfunktion, die `emerge` selbst bereitstellt. Mit der Option `--search` (bzw. `-s`) lassen sich die Paketnamen ebenfalls nach Mustern durchsuchen, allerdings mit deutlich hilfreicheren Ergebnissen als `ls` sie liefert. Suchen wir also auch hier in der Kategorie `app-editors` nach Paketen, die mit dem Buchstaben `v` beginnen. Die zu durchsuchende Kategorie wird mit dem Zeichen `@` markiert:

```
gentoo ~ # emerge --search "@app-editors/v"
Searching...
[ Results for search key : app-editors/v ]
[ Applications found : 3 ]

*  app-editors/vile
      Latest version available: 9.4d
      Latest version installed: [ Not Installed ]
      Size of files: 1,618 kB
      Homepage:      http://www.clark.net/pub/dickey/vile/vile.html
      Description:   VI Like Emacs -- yet another full-featured vi clone
      License:       GPL-2

*  app-editors/vim
      Latest version available: 7.0.174
      Latest version installed: [ Not Installed ]
      Size of files: 6,232 kB
      Homepage:      http://www.vim.org/
      Description:   Vim, an improved vi-style text editor
      License:       vim

*  app-editors/vim-core
      Latest version available: 7.0.174
      Latest version installed: [ Not Installed ]
      Size of files: 6,232 kB
      Homepage:      http://www.vim.org/
      Description:   vim and gvim shared files
      License:       vim
```

Die drei bereits mit `ls` ermittelten Pakete beschreibt `emerge` deutlich detaillierter, und insbesondere die `Description` liefert klare Aussagen über den Inhalt eines Pakets.

13.2 esearch

Gehen wir einen Schritt weiter und suchen ein Paket, dessen Namen wir – anders als im `vim`-Beispiel – nicht einmal ungefähr kennen, beispielsweise eines, das die Funktionalität von `emerge --search` bietet, aber schneller ist.

Wir wollen dafür vorzugsweise die Beschreibung (also das `Description`-Feld) der Ebuilds durchsuchen. `emerge` bietet diese Funktion über die Option `--searchdesc` (bzw. `-S`); das Problem bei dieser Operation liegt jedoch in der Geschwindigkeit, denn `emerge --searchdesc` ist wirklich langsam, so dass eine schnellere Alternative tatsächlich wünschenswert ist.

Suchen wir also einen Ersatz für das Kommando `emerge --search`:

```
gentoo ~ # emerge -S "emerge --search"
Searching...
[ Results for search key : emerge --search ]
[ Applications found : 1 ]

*  app-portage/esearch
      Latest version available: 0.7.1
      Latest version installed: [ Not Installed ]
      Size of files: 10 kB
      Homepage:      http://david-peter.de/esearch.html
      Description:   Replacement for 'emerge --search' with search-index
      License:       GPL-2
```

Welch glücklicher – zugegebenermaßen leicht konstruierter – Zufall: Es gibt das Paket `esearch`, das auf diese Suchoperationen spezialisiert ist und deutlich schneller als Portage arbeitet. Installieren wir das Tool:

```
gentoo ~ # emerge -av app-portage/esearch

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-portage/esearch-0.7.1  LINGUAS="-it" 11 kB

Total: 1 package (1 new), Size of downloads: 11 kB

Would you like to merge these packages? [Yes/No] Yes
...
```

Damit stehen drei Kommandozeilen-Tools für die Paket-Suche zur Verfügung: `esearch`, `eupdatedb` und `esync`. Die Beschleunigung des Suchbefehls über `esearch` resultiert im Wesentlichen aus der Verwendung eines Suchindex. `eupdatedb` dient dessen Erstellung/Update. Der Vorgang dauert zwar eine gewisse Zeit, aber dafür sind die nachfolgenden Operationen umso schneller.

Da ein Update der Datenbank nur erforderlich ist, wenn wir zuvor die Paketdefinitionen des Portage-Baums mit `emerge --sync` auf den neuesten Stand gebracht haben, kombiniert der Aufruf `esync` beide Operationen (`emerge --sync` mit anschließendem `eupdatedb`). Wie wir diesen Schritt automatisieren, beschreibt Kapitel 16.6.4 ab Seite 360.

An dieser Stelle genügt uns das erstmalige Erstellen der Datenbank:

```
gentoo ~ # eupdatedb
 * indexing: 0 ebuilds to go
 * esearch-index generated in 6 minute(s) and 47 second(s)
 * indexed 11562 ebuilds
 * size of esearch-index: 1849 kB
```

Wir ersetzen `emerge -S` durch `esearch -S` und erhalten das gleiche Ergebnis, aber deutlich schneller:

```
gentoo ~ # esearch -S "emerge --search"
[ Results for search key : emerge --search ]
[ Applications found : 1 ]

*  app-portage/esearch
      Latest version available: 0.7.1
      Latest version installed: 0.7.1
      Size of downloaded files: 30 kB
      Homepage:    http://david-peter.de/esearch.html
      Description: Replacement for 'emerge --search' with search-index
      License:     GPL-2
```

Der `esearch`-Befehl alleine ersetzt den Aufruf `emerge --search` und sucht ausschließlich in Paketnamen. Die Option `--searchdesc` (bzw. `-S`), die wir hier verwendet haben, entspricht `emerge --searchdesc` und durchsucht die Paketbeschreibungen.

Um die Kategorie anzugeben, haben wir bei `emerge --search` das Zeichen `@` vorangestellt. Den gleichen Effekt erzielen wir bei `esearch` über die Option `--fullname` (bzw. `-F`):

```
gentoo ~ # esearch -F "app-editors/v"
[ Results for search key : app-editors/v ]
[ Applications found : 3 ]

*  app-editors/vile
      Latest version available: 9.4d
      Latest version installed: [ Not Installed ]
      Size of files: 1,618 kB
      Homepage:      http://www.clark.net/pub/dickey/vile/vile.html
      Description:   VI Like Emacs -- yet another full-featured vi
      clone
      License:       GPL-2

*  app-editors/vim
      Latest version available: 7.0.174
      Latest version installed: [ Not Installed ]
      Size of files: 6,232 kB
      Homepage:      http://www.vim.org/
      Description:   Vim, an improved vi-style text editor
      License:       vim
```

```
*  app-editors/vim-core
      Latest version available: 7.0.174
      Latest version installed: [ Not Installed ]
      Size of files: 6,232 kB
      Homepage:      http://www.vim.org/
      Description:   vim and gvim shared files
      License:       vim
```

Bei der Suche mit `esearch` lassen sich auch reguläre Ausdrücke verwenden:

```
gentoo ~ # esearch "^vim$"
[ Results for search key : app-editors/v ]
[ Applications found : 1 ]

*  app-editors/vim
      Latest version available: 7.0.174
      Latest version installed: [ Not Installed ]
      Size of files: 6,232 kB
      Homepage:      http://www.vim.org/
      Description:   Vim, an improved vi-style text editor
      License:       vim
```

Hier suchen wir z. B. nach einem Paket, das genau `vim` heißt. Um die Suche auf bereits installierte Pakete zu reduzieren, dient die Option `-I` (bzw. `--instonly`).

13.3 Fortgeschrittene Optionen für esearch

Auch das Ausgabeformat von `esearch` lässt sich gut beeinflussen. Bei besonders zahlreichen Treffern hilft z. B. die Option `--compact` (bzw. `-c`) den Überblick zu bewahren:

```
gentoo ~ # esearch -c -F "app-editors/v"
[ N] app-editors/vile (9.4d):  VI Like Emacs -- yet another full-featured
 vi clone
[ N] app-editors/vim (7.1.042):  Vim, an improved vi-style text editor
[ N] app-editors/vim-core (7.1.042):  vim and gvim shared files
```

Über der Switch `--verbose` (bzw. `-v`) erhält man umgekehrt detailliertere Informationen über die gefundenen Pakete:

```
gentoo ~ # esearch -v "^vim$"
*  app-editors/vim
      Latest version available: 7.1.042
      Latest version installed: [ Not Installed ]
      Unstable version:         7.1.087
```

```
      Use Flags (stable):       -acl -bash-completion -cscope -gpm
      -minimal -nls -perl -python -ruby -vim-pager -vim-with-x
      Size of downloaded files: [no/bad digest]
      Homepage:    http://www.vim.org/
      Description: Vim, an improved vi-style text editor
      License:     vim
```

Hier erfährt man beispielsweise etwas über die verfügbaren USE-Flags und eventuell instabile Versionen. `--nocolor` (bzw. `-n`) unterdrückt übrigens die Farbe in der Ausgabe.

Eher für Entwickler interessant dürften die Optionen `--ebuild` und `--own` sein: `--ebuild` (bzw. `-e`) zeigt die Ebuilds der verfügbaren Paketversionen an:

```
gentoo ~ # esearch -e -F "editors/vim"
[ N] app-editors/vim (7.1.042):  Vim, an improved vi-style text editor
 Portage [1] vim-6.4
 Portage [2] vim-7.0.174
 Portage [3] vim-7.0.235
 Portage [4] vim-7.1
 Portage [5] vim-7.1-r1
 Portage [6] vim-7.1.002
 Portage [7] vim-7.1.028
 Portage [8] vim-7.1.042
 Portage [9] vim-7.1.087

[ N] app-editors/vim-core (7.1.042):  vim and gvim shared files
 Portage [10] vim-core-6.4
 Portage [11] vim-core-7.0.174
 Portage [12] vim-core-7.0.235
 Portage [13] vim-core-7.1
 Portage [14] vim-core-7.1-r1
 Portage [15] vim-core-7.1.002
 Portage [16] vim-core-7.1.028
 Portage [17] vim-core-7.1.042
 Portage [18] vim-core-7.1.087

Show Ebuild:
```

Den ausgewählten Ebuild öffnet `esearch` dann mit dem in `/etc/rc.conf` gewählten Editor.

Die Option `--own` (bzw. `-o`) gibt die Möglichkeit, die Ausgabe der Suchergebnisse zu „formatieren“:

```
gentoo ~ # esearch -o "%c/%n\n" -F "editors/vim"
app-editors/vim
app-editors/vim-core
```

Hier steht `%c` für die Kategorie des gefundenen Paketes, `%n` für den Paketnamen. Über weitere Formatcodes gibt bei Bedarf die Man-Page Auskunft.

13.4 Suchen mit qsearch und qgrep

Wer die Aktualisierung der `esearch`-Datenbank nach jeder Synchronisation des Portage-Baums als zu umständlich empfindet, dem seien `qsearch` und `qgrep` aus dem Paket `app-portage/portage-utils` (siehe auch Kapitel 11.2) empfohlen. Beide sind nicht ganz so schnell wie `esearch`, aber immer noch deutlich schneller als `emerge --search`, da sie in C implementiert sind.

`qsearch` beherrscht die Suche in Paketnamen und Paketbeschreibungen. Standardmäßig wird im Paketnamen gesucht:

```
gentoo ~ # qsearch esearch
app-portage/esearch Replacement for 'emerge --search' with search-index
app-vim/multiplesearch vim plugin: allows multiple highlighted searches
```

Die Ausgabe ist knapp gehalten und angenehm übersichtlich. In der Beschreibung suchen wir wie bei `esearch` mit der Option `--desc` (bzw. `-S`):

```
gentoo ~ # qsearch -S "emerge --search"
app-portage/esearch Replacement for 'emerge --search' with search-index
```

Bei der Ausgabe haben wir nicht ganz so viel Gestaltungsfreiheit wie bei `esearch`. Wir können die Anzeige der Paketbeschreibung mit der Option `--name-only` (bzw. `-N`) entfernen und uns so auf den Paketnamen beschränken oder über `--homepage` (bzw. `-H`) nur die Homepage des Projekts ausgeben lassen:

```
gentoo ~ # qsearch -H -S "emerge --search"
app-portage/esearch http://david-peter.de/esearch.html
```

Das zweite Suchwerkzeug aus `app-portage/portage-utils` heißt `qgrep` und ist auch eher ein Werkzeug für Entwickler – darum nur einige kurze Bemerkungen dazu.

`qgrep` durchsucht nicht nur die Beschreibung der Ebuilds, sondern gleich die gesamt Ebuild-Datei:

```
gentoo ~ # qgrep -N " esearch"
app-portage/esearch-0.7.1-r2:doexe eupdatedb.py esearch.py esync.py comm
on.py || die "doexe failed"
app-portage/esearch-0.7.1-r3:doexe eupdatedb.py esearch.py esync.py comm
on.py || die "doexe failed"
app-portage/esearch-0.7.1-r4:doexe eupdatedb.py esearch.py esync.py comm
on.py || die "doexe failed"
app-portage/esearch-0.7.1:doexe eupdatedb.py esearch.py esync.py common.
py || die "doexe failed"
```

Wir haben hier z. B. nach `esearch` mit einem vorangestellten Leerzeichen gesucht. Die Option `--with-name` (bzw. `-N`) liefert bei jedem Treffer den zugehörigen Paketnamen. Das Resultat sind einige Zeilen aus den `esearch`-Ebuilds.

In den meisten Fällen ist die Suche mit Hilfe von `qsearch` (oder `esearch`) effektiver. Aber wenn man das gewünschte Schlagwort eher im Code bzw. einer Kommentarzeile versteckt erwartet, empfiehlt sich `qgrep`.

13.5 Paketsuche mit eix

`eix` ist das Monster unter den Such-Programmen für Gentoo. Schon ein Blick in die Man-Page offenbart, dass man hier etwas mehr Funktionalität erwarten darf.

Wir wollen uns hier nur mit den Grundfunktionen beschäftigen und die wichtigsten herausgreifen, um interessante Software für unser System zu finden. Wir kommen auf das Werkzeug im nächsten Kapitel ab Seite 316 noch einmal zu sprechen, denn gerade im Bereich Overlays bietet es einige hilfreiche Eigenschaften.

Installieren wir `eix` also zunächst einmal:

```
gentoo ~ # emerge -av app-portage/eix

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-portage/eix-0.7.9  USE="-sqlite" 348 kB

Total: 1 package (1 new), Size of downloads: 348 kB

Would you like to merge these packages? [Yes/No] Yes
```

Hinweis für Systeme mit Netzwerkanbindung

Sie benötigen eine funktionieren Netzwerkverbindung, um das Paket hier zu installieren.

`eix` arbeitet, ebenso wie `esearch`, zur Beschleunigung mit einem Cache, ohne den es auch gar nicht funktioniert:

```
gentoo ~ # eix
Can't open the database file /var/cache/eix for reading (mode = 'rb')
Did you forget to create it with 'update-eix'?
```

Den Cache erstellen wir erstmalig mit `update-eix`:

```
gentoo ~ # update-eix
Reading Portage settings ..
Building database (/var/cache/eix) ..
[0] "gentoo" /usr/portage/ (cache: metadata)
     Reading 100%
Applying masks ..
Database contains 12804 packages in 151 categories.
```

Danach lässt sich `eix` genauso wie `esearch` verwenden und liefert die gleichen Ergebnisse in einem etwas komplexeren Ausgabeformat:

```
gentoo ~ # eix "^vim$"
[U] app-editors/vim
     Available versions:  6.4 7.0.235 7.1.042 7.1.123 ~7.1.164 ~7.1.213
                          {acl bash-completion cscope gpm minimal nls
                           perl python ruby vim-pager vim-with-x}
     Installed versions:  7.0.17(00:13:39 17.10.2006)
                          (bash-completion perl -acl -cscope -gpm
                           -minimal -mzscheme -nls -python -ruby
                           -vim-pager -vim-with-x)
     Homepage:            http://www.vim.org/
     Description:         Vim, an improved vi-style text editor
```

Das einleitende `U` zeigt an, wie von `emerge` gewohnt, dass wir das Paket aktualisieren könnten. Außerdem zeigt `eix` alle verfügbaren Versionen inklusive USE-Flags der aktuellsten Version an. Instabile Versionsnummern sind mit einer führenden `~` versehen. Die aktuell installierte Version wird mit dem Installationszeitpunkt und den gewählten USE-Flags dargestellt. Die übrigen Angaben entsprechen jenen von `esearch`.

Mit der Option `--installed` (bzw. `-I`) beschränken wir die Suche wie auch schon bei `esearch` auf die installierten Pakete. `--compact` (bzw. `-c`) liefert ein kompakteres Format:

```
gentoo ~ # eix -I -c eix
[I] app-portage/eix (0.10.2@13.01.2008): Small utility for searching
ebuilds with indexing for fast results
```

Schauen wir uns nun noch einige Suchvarianten an: In den Beschreibungen suchen wir in gewohnter Weise mit der Option `-S` (bzw. `--description`):

```
gentoo ~ # eix -c -S "emerge --search"
[I] app-portage/esearch (0.7.1@17.12.2007): Replacement for 'emerge
--search' with search-index
```

Darüber hinaus bietet `eix` speziellere Suchmöglichkeiten, so z. B. die Suche im URL-Pfad der Projekt-Homepage. Die Option ist `--homepage` (bzw. `-H`):

```
gentoo ~ # eix -H "eix"
[I] app-portage/eix
     Available versions:  0.9.9 0.9.10 0.10.2 ~0.10.3 {sqlite}
     Installed versions:  0.10.2(11:24:36 13.01.2008)(-sqlite)
     Homepage:            http://eix.sourceforge.net
     Description:         Small utility for searching ebuilds with
                          indexing for fast results
```

Nach Kategorie lässt sich über die Option `--category` (bzw. `-C`) suchen, hier demonstriert am Beispiel der Kategorie `app-antivirus`:

```
gentoo ~ # eix -c -C antivirus
[N] app-antivirus/bitdefender-console (7.0.1-r1): BitDefender console
    antivirus
[N] app-antivirus/clamav (0.91.2-r1): Clam Anti-Virus Scanner
[N] app-antivirus/f-prot (4.6.7): Frisk Software's f-prot virus
    scanner
[N] app-antivirus/klamav (0.41): KlamAV is a KDE frontend for the
    ClamAV antivirus.
```

Während `eix` standardmäßig nur im Namen des Paketes sucht, lässt sich die Suche mit `--category-name` (bzw. `-A`) um den Namen der Kategorie erweitern. Mit `eix -A app-editors/v` würden wir also die gleiche Ausgabe erhalten wie mit dem Befehl `esearch -F "app-editors/v"`.

Nützlich ist auch die Suche nach bestimmten USE-Flags, die wir mit der Option `--use` (bzw. `-U`) aktivieren. Ähnliches vermag aber natürlich auch `euses`, das wir schon auf Seite 115 beschrieben haben.

`eix` bietet eine ganze Reihe von Optionen und Variationsmöglichkeiten, die wir nicht alle darstellen können. Wir kommen im nächsten Kapitel noch einmal auf einige Varianten im Zusammenhang mit Overlays zu sprechen und schließen hier mit einer letzten nützlichen Suchoption: `--fuzzy` (bzw. `-f`). Sie erlaubt eine unscharfe Suche, so dass Treffer möglich sind, auch wenn wir uns bei Paketnamen oder Schlagwort nicht ganz sicher sind:

```
gentoo ~ # eix -I -c -f 2 eiks
[I] app-portage/eix (0.10.2@13.01.2008): Small utility for searching
ebuilds with indexing for fast results
[I] www-client/elinks (0.11.3@10.12.2007): Advanced and
well-established text-mode web browser
Found 2 matches.
```

Die Zahl hinter `-f` gibt an, wie viele Buchstaben im Suchstring falsch sein dürfen. Ohne Angabe akzeptiert `eix` zwei Fehler bei der Suche.

13.6 Weitere Möglichkeiten

Die vorgestellten Kommandozeilenwerkzeuge sind, wie gesagt, nur eine Möglichkeit, Pakete aufzufinden. Wer eine webbasierte Alternative sucht, dem sei `http://packages.gentoo.org` empfohlen. Diese Seite stellt die gefundenen Pakete deutlich ansprechender dar als die Kommandozeilen-Tools dies können.

Häufig findet man über die Suche im Internet auch Software, die man zwar installieren möchte, nach der die genannten Werkzeuge jedoch vergeblich suchen; das bedeutet, dass Portage selbst kein entsprechendes Paket bereitstellt. Das heißt aber nicht zwingend, dass es ein solches Paket nicht gibt, denn es ist durchaus möglich, dass irgendjemand schon eine Paketdefinition, also einen *Ebuild* für diese Software geschrieben hat.

Folgende alternativen Schritte helfen, über den Portage-Baum hinaus eine Paketdefinition zu finden:

- Suche in der Bug-Datenbank `http://bugs.gentoo.org`. Diese fördert häufig Ebuilds zu Tage, die noch nicht im Portage-Baum enthalten sind. Wie sich solche Ebuilds in das eigene System einbinden lassen, beleuchten wir in den Kapiteln 14 und 15.
- In den unter `http://overlays.gentoo.org` gelisteten Overlays finden sich zahlreiche experimentelle Ebuilds. Hier sei vor allem das Projekt *Sunrise* erwähnt.[1] Auch dazu mehr im nächsten Kapitel.
- Natürlich lässt sich auch das Gentoo-Forum[2] durchsuchen. Es gibt eine eigene Sektion **Unsupported Software**, die sich Paketen außerhalb des Portage-Baums widmet.
- Weiß man wirklich gar nicht mehr weiter und benötigt kurzfristig einen Hinweis ist auch IRC (*Internet Relay Chatting*) zu empfehlen.[3] Unter den Tipps gehen wir ab Seite 370 etwas genauer darauf ein.

Eines sollte man bei diesen Maßnahmen allerdings nie vergessen: Wenn ein Paket im Portage-Baum fehlt, dann bedingt das im Normalfall auch eine deutlich geringere Qualität der Paketdefinition. Mit solchen Ebuilds sollte man also vorsichtig sein. Aber auch auf diesen Aspekt gehen wir genauer im nächsten Kapitel ein.

[1] `http://overlays.gentoo.org/proj/sunrise`
[2] `http://forums.gentoo.org`
[3] Kanal `#gentoo` auf `irc.freenode.net`

Kapitel

14 Gentoo erweitern

Der Portage-Baum liefert derzeit rund 12.000 Pakete, also eine durchaus beachtliche Auswahl. Das ändert aber nichts daran, dass man gelegentlich eine Software finden wird, die eben noch nicht im Portage-Baum verfügbar ist, denn die Kapazität der Entwickler ist begrenzt.

Wir wollen im Folgenden beschreiben, wie sich das eigene System recht unproblematisch mit experimentellen Paketen und der neuesten Software ausstatten lässt. Es wäre allerdings fahrlässig, den Umgang mit experimentellen Paketen als trivial und risikolos zu beschreiben; darum gehen wir noch einmal kurz auf die *Qualität* der Ebuilds ein.

Wir hatten uns schon in Kapitel 5.4 mit „stabilen“ und „instabilen“ Paketen beschäftigt. Ist eine Paket-Version als „instabil“ markiert, treffen die Entwickler mit dieser Markierung eine deutliche Qualitätsaussage. Solche Pakete wurden noch nicht ausreichend getestet, um Inkompatibilitäten auszuschließen, und können kritische Fehler enthalten.

Als Nutzer können wir bewusst diese Qualitätsaussage der Entwickler außer Acht lassen und auf der Grundlage eines stabilen Systems einzelne Pakete

über die Datei /etc/portage/package.keywords auswählen, die Portage dann in der instabilen Version akzeptiert. Man sollte das aber nur bei Paketen tun, bei denen die instabile Version Features bietet, die man tatsächlich benötigt. Das setzt allerdings ein wenig Beschäftigung mit diesem Paket voraus, und daraus resultieren zwei Vorteile: Man kann mit auftretenden Problemen besser umgehen, aber auch qualifizierte Fehlerberichte liefern, was wiederum die Entwickler freut und ihnen die Behebung des Problems erleichtert.

Nun gibt es über den Mechanismus des *Keywordings* hinaus eine weitere Abstufung, um experimentelle Pakete in das System einzubinden: *Overlays*. Über diese lassen sich sehr experimentelle Pakete einbinden, die nicht einmal von den Gentoo-Entwicklern selbst stammen müssen; darum ist hier der Warnhinweis in Bezug auf deren Qualität noch eindringlicher. Der Qualitätsunterschied zu den Paketen des Portage-Baumes ist bei den Overlays nicht zu unterschätzen!

Auf der anderen Seite arbeiten oft Personen mit großem Fachwissen in einem spezifischen Bereich an diesen Overlays, und so sind diese Projekte ein Ort regen Austauschs und wertvoller Ideen, die nach einiger Zeit des Testens in den Portage-Baum einfließen. Schauen wir uns aber nun einmal an, wie diese Overlays überhaupt funktionieren.

14.1 Overlays

Portage ermöglicht über die Variable PORTAGE_OVERLAYS in der Datei /etc/make.conf die Einbindung einer beliebigen Menge an Dateibäumen, die den Original-Portage-Baum „überdecken“ (daher der Begriff „Overlay“).

Ein Overlay ist genauso strukturiert wie der normale Portage-Baum (siehe Seite 41). Einzelne Verzeichnisse entsprechen einer Kategorie, und innerhalb dieser Kategorien befinden sich Ordner, die jeweils einem einzelnen Paket entsprechen und die Ebuilds enthalten.

Hier beispielhaft die Struktur des Overlays php-experimental, wenn man es auf der eigenen Maschine installiert; wie das vor sich geht, beschreiben wir im Laufe dieses Kapitels:

```
gentoo ~ # ls /usr/portage/local/layman/php-experimental
app-admin
dev-db
dev-lang
dev-php
dev-php4
dev-php5
eclass
profiles
```

```
gentoo ~ # ls /usr/portage/local/layman/php-experimental/dev-php5
ffmpeg-php
pecl-uuid
```

Das Overlay enthält also weniger Kategorien, und auch jede Kategorie enthält (wie am Beispiel `dev-php5` demonstriert) nur eine reduzierte, spezialisierte Anzahl an Paketen. Abgesehen davon ist ein Overlay strukturell aber mit dem Portage-Baum vergleichbar (siehe Kapitel 1.7.1) und kann sogar eigene Eclasses (siehe Seite 325) oder Profil-Informationen liefern (siehe Kapitel 5.2). Das PHP-Overlay enthält Eclasses in dem Verzeichnis `eclass` und Profilinformationen in `profiles`.

Das Overlay befindet sich unter dem Pfad `/usr/portage/local/layman/php-experimental`. Möchte man es nutzen, fügt man diesen Pfad der Variable `PORTAGE_OVERLAYS` hinzu, die mehrere, durch Leerzeichen getrennte Pfade enthalten enthalten kann.

Portage selbst betrachtet Overlays als normale Paketbäume, die den Standard-Paketbaum ergänzen bzw. überdecken. Dieser Mechanismus und die Tatsache, dass Abhängigkeiten zwischen Gentoo-Paketen im Vergleich mit anderen Distributionen deutlich geringer sind, erlaubt es, das Grundgerüst der Distribution recht einfach zu erweitern.

Wie wir ein eigenes Overlay erstellen und einbinden, zeigen wir im nächsten Kapitel ab Seite 321. Hier wollen wir uns aber erst einmal damit beschäftigen, fremde Overlays in unser System zu integrieren.

14.2 overlays.gentoo.org

Ursprünglich war der Overlays-Mechanismus eher für Entwickler gedacht, um experimentelle Versionen ihrer Pakete außerhalb des Portage-Baumes zu pflegen und zu testen. Recht schnell haben dann einige Entwickler begonnen, ihre Overlays über das Netz zu exportieren und so den Nutzern zur Verfügung zu stellen. Dieser Schritt ist für beide Seiten von Vorteil: Die Nutzer haben plötzlich die Möglichkeit, die neueste Software innerhalb des gewohnten Gentoo-Paket-Managements zu testen. Gleichzeitig erhalten Entwickler wertvolles Feedback von Anwendern, die sich ja bewusst für experimentelle Pakete entscheiden und meist auch über das Know-how verfügen, mit auftretenden Fehlern umzugehen.

Als der Overlay-Gedanke immer mehr Anklang fand und mehr und mehr Overlays im Netz auftauchten, wurde es sinnvoll, diese Anstrengungen zu zentralisieren. Dies hat zu dem Projekt `overlays.gentoo.org` geführt[1], das jedem Entwickler die Möglichkeit gibt, sein eigenes Overlay zu verwalten. Auch jedes Gentoo-Unterprojekt kann sein eigenes Overlay anlegen.

[1] `http://overlays.gentoo.org`

Jedes Overlay wird über eine `trac`-Installation verwaltet, die automatisch einen Source-Code-Viewer und ein Wiki zu Dokumentationszwecken zur Verfügung stellt. Alle Veränderungen an den Overlays werden als zentraler RSS-Feed zusammengefasst und auch direkt auf `overlays.gentoo.org` angezeigt. Gerade Dokumentation und der einfache Zugriff auf Source-Code und ChangeLog sind bei diesen experimentellen Paketen besonders wichtig.

Doch `overlays.gentoo.org` ist nicht die einzige Quelle für experimentelle Pakete. Es gibt viele langjährige Gentoo-Nutzer, die an anderen Stellen eigene Overlays pflegen. Viele davon sind über `layman` (siehe den folgenden Abschnitt) verfügbar, aber einige muss man im Netz suchen, wobei die Gentoo-Foren oder das Gentoo-Wiki gute Anlaufstellen sind.

Hinweis für Systeme mit Netzwerkanbindung

Overlays lassen sich nur direkt über das Netz herunterladen. Sie benötigen also für die folgenden Abschnitte eine funktionierende Netzwerkverbindung.

14.3 layman

Das Skript `layman` vereinfacht die Benutzung von Overlays. Es ist in der Lage, Overlays automatisch im System zu installieren und sie zu verwalten. Darum wollen wir es zunächst installieren:

```
gentoo ~ # emerge -av app-portage/layman

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] net-misc/neon-0.26.1-r1  USE="nls ssl zlib -expat -socks
5" 0 kB
[ebuild  N    ] dev-util/subversion-1.3.2-r3  USE="apache2 berkdb nls pe
rl python zlib -bash-completion -emacs -java -nowebdav -ruby" 0 kB
[ebuild  N    ] app-portage/layman-1.0.6  0 kB

Total: 3 packages (3 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
```

Zum Abschluss der Installation gibt der Ebuild die weiteren Schritte vor:

```
 * You are now ready to add overlays into your system.
 *
```

```
 * layman -L
 *
 * will display a list of available overlays.
 *
 * Select one and add it using
 *
 * layman -a overlay-name
 *
 * If this is the very first overlay you add with layman, you need
 * to append the following statement to your /etc/make.conf file:
 *
 * source /usr/portage/local/layman/make.conf
 *
```

Es gibt eine zentrale Liste bekannterer Overlays[2], die von den Entwicklern regelmäßig aktualisiert wird. Über die Option `--list` (bzw. `-L`) lädt `layman` diese Liste herunter und zeigt die verfügbaren Overlays an:

```
gentoo ~ # layman -L
...
* chtekk-apps       [Subversion] (source: http://overlays.gentoo.org...)
* chtekk-syscp      [Subversion] (source: http://overlays.gentoo.org...)
* common-lisp       [Darcs     ] (source: http://www.common-lisp.net...)
* dev-zero          [Subversion] (source: http://overlays.gentoo.org...)
* efika             [Subversion] (source: http://overlays.gentoo.org...)
* genstef           [Subversion] (source: http://overlays.gentoo.org...)
* gentopia          [Subversion] (source: http://overlays.gentoo.org...)
...
```

Diese Liste führt den Namen des Overlays, seinen Typ, der im Normalfall dem Namen des eingesetzten Revisionskontrollsystems entspricht, und schließlich den Link auf die Quelle des Overlays. Die meisten Overlays werden in einem `subversion`-Repository bereitgehalten, sind also vom Typ `Subversion`. Layman beherrscht aber neben `subversion` alle anderen verbreiteten Versionskontrollsysteme und erlaubt auch die Installation aus `tar`-Archiven.

Reicht der kurze Name für eine Beschreibung nicht aus, können wir den Output über den bekannten `--verbose`-Schalter (bzw. `-v`) erhöhen. Damit spuckt `layman` allerdings eine unangenehm lange Liste aus. Es ist dann besser, sich mit der Option `--info` (bzw. `-i`) in Kombination mit dem Namen des Overlays die spezifische Beschreibung für ein einzelnes Overlay anzeigen zu lassen. Allerdings wird diese Option erst von neueren `layman`-Versionen unterstützt:

```
gentoo ~ # layman -i webapps-experimental
* webapps-experimental
```

[2] `http://www.gentoo.org/proj/en/overlays/layman-global.txt`

```
* ~~~~~~~~~~~~~~~~~~~~
* Source  : https://overlays.gentoo.org/svn/proj/webapps/experimental
* Contact : web-apps@gentoo.org
* Type    : Subversion; Priority: 50
*
* Description:
*   This is the home of Gentoo's wider collection of ebuilds for
*   web-based applications. This is where we collect all the ebuilds
*   submitted to Bugzilla by our users, and make them available in
*   an easy-to-use overlay for wider testing.
*
* Link:
*
*   http://overlays.gentoo.org
```

Mit der Option `--add` (bzw. `-a`) installiert `layman` dieses Overlay:

```
gentoo ~ # layman -a webapps-experimental
```

`layman` lädt das Overlay nun automatisch über `subversion` herunter und legt es in dem in `/etc/layman/layman.cfg` unter `storage` angegebenen Ort ab:

```
storage   : /usr/portage/local/layman
```

Der Name des dort angelegten Unterverzeichnisses entspricht der Bezeichnung des Overlays:

```
gentoo ~ # ls -la /usr/portage/local/layman
total 41
drwxr-xr-x  3 root root   264 2006-10-21 18:53 .
drwxr-xr-x  3 root root    72 2006-09-17 16:03 ..
-rw-r--r--  1 root root 26314 2006-10-21 18:53 cache_65bd38402ac8431067b5
4904bd2ed2d1.xml
-rw-r--r--  1 root root    69 2006-10-21 18:53 make.conf
-rw-r--r--  1 root root   412 2006-10-21 18:53 overlays.xml
drwxr-xr-x 15 root root   496 2006-10-21 18:54 webapps-experimental
```

Die drei anderen Dateien hat `layman` für die Verwaltung der Overlays angelegt. `overlays.xml` enthält die Liste der bereits installierten Overlays, während die `cache_*.xml`-Datei die zentrale Liste zwischenspeichert, damit `layman` diese Daten nicht für jede Operation erneut herunterladen muss.

Die Datei, die uns hier aber eigentlich interessiert, ist `make.conf`, die entsprechend der zentralen Gentoo-Konfigurationsdatei `/etc/make.conf` benannt ist. Sie enthält nur eine Variable, die normalerweise auch in `/etc/make.conf` vorkommt:

```
gentoo ~ # cat /usr/portage/local/layman/make.conf
PORTDIR_OVERLAY="$PORTDIR_OVERLAY
/usr/portage/local/layman/webapps-experimental
"
```

Um dem System Overlays automatisiert hinzuzufügen bzw. zu deinstallieren, muss `layman` die Variable `PORTDIR_OVERLAY` automatisiert verwalten. Dies geschieht in der ausgelagerten Datei `/usr/portage/local/layman/make.conf`, nicht direkt in `/etc/make.conf`, da eine automatisierte Bearbeitung solch zentraler Konfigurationsdateien wenig wünschenswert ist.

Natürlich müssen wir die Erweiterung in die Datei `/etc/make.conf` einbinden:

```
gentoo ~ # echo "source /usr/portage/local/layman/make.conf"
> >> /etc/make.conf
```

Der `source`-Befehl bewirkt, dass Portage die externe Konfigurationsdatei einliest und auswertet. Da die von `layman` verwaltete Konfigurationsdatei die Pfade der mit dem Tool installierten Overlays nur an die Variable `PORTDIR_OVERLAY` anhängt, kann der Nutzer andere Overlays auch weiterhin manuell über die Standardeinstellung in `/etc/make.conf` verwalten. Damit ist `layman` vollständig eingerichtet, und wir können es für das Management von Overlays verwenden.

Overlays entfernen wir mit der Option `--delete` (bzw. `-d`) wieder aus dem System:

```
gentoo ~ # layman -d webapps-experimental
```

Die häufigste Operation wird allerdings das Update der Overlays sein, und zwar über die Option `--sync-all` (bzw. `-S`). Damit synchronisieren wir alle installierten Overlays nacheinander. Mit der Option `--sync` (bzw. `-s`) und der Angabe eines Overlay-Namens lassen sich einzelne Overlays synchronisieren.

```
gentoo ~ # layman -s webapps-experimental
```

Die derzeit installierten Overlays sind mit dem Flag `--list-local` (bzw. `-l`) abrufbar:

```
gentoo ~ # layman -l
* webapps-experimental     [Subversion] (source: http://overlays.gentoo.
org...)
```

14.4 Overlays mit eix durchsuchen

Da es eine ganze Reihe zusätzlicher Pakete in den Overlays gibt, wäre es praktisch, diese in die Suche einzubeziehen, die wir im Kapitel 13 besprochen haben. `qsearch` kann aber beispielsweise mit Overlays gar nichts anfangen. `emerge --search` bzw. `esearch` beziehen zumindest die lokal installierten Overlays in die Suche ein. Für die Suche über diese Programme wären wir also gezwungen, unserem System ein Overlay hinzuzufügen, damit wir überhaupt Pakete darin finden.

`eix` bietet als einziges Werkzeug die Möglichkeit *virtuelle*, also externe Overlays in die Suche einzubeziehen, auch ohne dass diese lokal installiert sind. Allerdings müssen wir dafür die Konfiguration des Programms anpassen, denn per Default durchsucht auch `eix` nur lokal installierte Overlays, wie ein `update-eix` zeigt:

```
gentoo ~ # update-eix
Reading Portage settings ..
Building database (/var/cache/eix) ..
[0] "gentoo" /usr/portage/ (cache: metadata)
     Reading 100%
[1] "wrobel" /usr/portage/local/layman/wrobel (cache: none)
     Reading 100%
[2] "kolab" /usr/portage/local/layman/kolab (cache: none)
     Reading 100%
[3] "webapp-experimental"
/usr/portage/local/layman/webapps-experimental (cache: none)
     Reading 100%
[4] "sunrise" /usr/portage/local/layman/sunrise (cache: none)
     Reading 100%
Applying masks ..
Database contains 12826 packages in 151 categories.
```

Als Beispiel nehmen wir einmal an, wir suchen einen Ebuild für das Spiel „Second Life“. Dieser befindet sich nicht direkt im Portage-Baum, und so findet `eix` bei der derzeitigen Konfiguration nichts:

```
gentoo ~ # eix -c secondlife
No matches found.
```

Um Informationen über Pakete in externen Overlays verfügbar zu machen, verwenden wir das Skript `update-eix-remote` mit der Aktion `update`:

```
gentoo ~ # update-eix-remote update
 * Fetching eix-caches.tbz2
--10:31:58--
http://dev.gentooexperimental.org/eix_cache/eix-caches.tbz2
           => ‘eix-caches.tbz2’
```

```
Auflösen des Hostnamen >dev.gentooexperimental.org<.... 81.93.240.53
Verbindungsaufbau zu
dev.gentooexperimental.org|81.93.240.53|:80... verbunden.
HTTP Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 241.618 (236K) [application/octet-stream]

100%[======================================================================>]
   %241.618        1.25M/s

10:31:59 (1.25 MB/s) - >eix-caches.tbz2< gespeichert [241618/241618]

 * Unpacking data
 * Calling update-eix
...
```

`eix` lädt ein Datenarchiv mit den Ebuild-Informationen von etwas mehr als einhundert verschiedenen Overlays herunter und fügt es seiner lokalen Datenbank hinzu.

Würden wir allerdings jetzt den Portage-Baum aktualisieren und anschließend `update-eix` laufen lassen, würde `eix` diese Daten wieder verwerfen, da es standardmäßig eben nur die lokal installierten Overlays in die Datenbank einbezieht. Das lässt sich ändern, indem wir `KEEP_VIRTUALS` in der Konfigurationsdatei `/etc/eixrc` suchen und von `false` auf `true` setzen:

```
...
# BOOLEAN
# Keep virtuals of the old cache file by adding corresponding entries
# implicitly to the values of ADD_OVERLAY and ADD_CACHE_METHOD
KEEP_VIRTUALS='true'
...
```

`eixrc` ist eine etwas überfrachtete Datei, die mit über zweihundert Konfigurationsvariablen aufwartet, aber `eix` ist eben derzeit das einzige Werkzeug, das externe Overlays in die Suche einbeziehen kann.

Nach dieser Veränderung erhält `update-eix` die heruntergeladenen Dateien, und wir können problemlos in den externen Overlays suchen. Schauen wir also, ob unsere Suche nach dem „SecondLife"-Paket diesmal erfolgreich ist:

```
gentoo ~ # eix secondlife
[N] games-rpg/secondlife [1] (~1.18.0.6!m): A 3D MMORPG virtual world
                             entirely built and owned by its residents
[N] games-rpg/secondlife-bin (~1.13.3.2!m[2] ~1.17.2.0[3]
                              ~1.17.2.0-r1!m[3] ~1.18.0.6!m[1]
                              ~1.18.0.6!m[3]): A 3D MMORPG virtual
                             world entirely built and owned by its
                             residents
[N] games-simulation/secondlife (~1.18.0.6[5] ~1.18.0.6-r1!m[5]
```

```
                                  ...
                                   ~1.18.6.0_rc!m[5] ~1.18.6.1_rc!m[5]):
                                  A 3D MMORPG virtual world entirely
                                  built and owned by its residents
[N] games-simulation/secondlife-bin (~1.13.1.6!m[5] ~1.13.2.11!m[5]
                                       ...
                                       ~1.18.6.76453_alpha!m[4]): A 3D
                                      MMORPG virtual world entirely built
                                      and owned by its residents
[1] (layman/arcon)
[2] (layman/drizzt-overlay)
[3] (layman/sabayon)
[4] (layman/secondlife)
[5] (layman/zugaina)

Found 4 matches.
```

Wir haben aus Gründen der Übersicht an einigen Stellen ein paar Versionsnummern aus der Ausgabe entfernt. Aber auch so verdeutlicht das Ergebnis schon ein Problem: Welcher ist nun der beste Ebuild? Welches Overlay sollen wir lokal installieren und verwenden?

Die Anbieter der verschiedenen Pakete scheinen sich auch nicht einig bei der Wahl der Kategorie. Wir kommen als Benutzer auf jeden Fall nicht umhin, uns die verschiedenen Anbieter genauer anzusehen. Am besten geht man mit `layman --info` die fünf verschiedenen Overlay-Namen (`arcon`, `drizzt-overlay`, `sabayon`, `secondlife` und `zugaina`) durch und schaut sich dann auf den angegebenen Projektseiten an, wie aktuell die Ebuilds sind.

Sowohl im `arcon`-Overlay als auch im `drizzt-overlay` findet sich nur eine einige Monate alte Version, und es gibt keinerlei Updates. Ob die Installation gelingen würde ist eher fraglich. `sabayon` und `zugaina` scheinen hingegen einigermaßen aktuelle Ebuilds zu liefern. Hier sollte die Chance auf eine erfolgreiche Installation höher sein.

Letztlich scheint aber `secondlife` die sinnvollste Wahl, nicht aufgrund des Namens, der ja keine Aussage über die Aktualität der angebotenen Pakete trifft, sondern weil es von `overlays.gentoo.org` angeboten und von einem Gentoo-Entwickler gepflegt wird.

Dennoch ist das natürlich keine Garantie dafür, dass die Installation gelingt. Das bleibt der Nachteil der Overlays: Niemand kann wirklich garantieren, dass die Qualität der angebotenen Ebuilds ausreicht, um auf jedem System einwandfrei zu laufen.

Kapitel 15

Ebuilds schreiben

Trotz der hohen Anzahl an Paketen im Portage-Baum und trotz der Erweiterung durch Overlays wird es gelegentlich Situationen geben, in denen man einfach keine Paketdefinition für eine Software findet. Manchmal hilft dann nur noch die Gentoo-Bug-Datenbank, in die Nutzer ebenfalls Ebuilds hochladen können, wenn sie möchten, dass die Gentoo-Entwickler diese in den Portage-Baum aufnehmen.

Lässt sich ein Ebuild aber weder in einem der Overlays noch der Gentoo-Bug-Datenbank finden, gibt es drei letzte Alternativen, die Software im eigenen System zu installieren: Abwarten, bis jemand einen Ebuild schreibt, die Software manuell installieren oder selbst einen Ebuild schreiben.

Abwarten ist in den seltensten Fällen eine Option; die manuelle Installation wird zwar meist funktionieren, hat aber den gravierenden Nachteil, dass man den gewohnten Rahmen des Paketmanagements verlässt.

Bleibt also nur selbst einen Ebuild zu schreiben. Was für andere Distributionen wohl mehr als ungewöhnlich ist, liegt bei Gentoo durchaus nahe, denn es ist eher eine „Entwickler-Distribution“: Seine Benutzer sind bereit, Soft-

ware zu kompilieren, und sie haben in der Regel keine Berührungsängste mit Source-Code. Damit ist auch die Bereitschaft vorhanden, Hand an den Source-Code einer Paketdefinition anzulegen.

Bei manchen Paketen ist das Schreiben des Ebuilds nahezu trivial und damit die Einstiegshürde für bestimmte Software-Kategorien sehr niedrig. Ein Beispiel werden wir uns gleich ansehen. Wohlgemerkt gilt diese Aussage wirklich nur für einen kleinen Teil aller Pakete. Die weitaus meisten Ebuilds sind durchaus komplex und benötigen einiges Wissen über die zu installierende Software bzw. Paketmanagement im Allgemeinen.

Im Gegensatz zu der für binäre Distributionen typischen Trennung zwischen Paketdefinition und Paket zählt bei Gentoo einzig und allein der Ebuild; er ist *die* Grundlage für Austausch und Diskussion. Vor allem für den Austausch liegen die Vorteile auf der Hand, denn meist reduziert sich der Ebuild ja auf eine kleine Textdatei. Das ist auch einer der Gründe, warum sich viele experimentelle Pakete in der Gentoo-Bug-Datenbank befinden. Und da ein Projekt wie Gentoo von der regen Beteiligung der Nutzergemeinde abhängt, werden wir uns an einem möglichst einfachen Beispiel anschauen, was einen Ebuild eigentlich ausmacht.

Keine Sorge, es geht nicht darum, Sie zum Gentoo-Entwickler auszubilden. Wir wollen Grundlagen vermitteln und Ihnen die Möglichkeit geben, bei einfachen Software-Installationen das Portage-System nicht zu verlassen. Zudem lassen sich auf diese Weise weitere Interna des Portage-Systems besser darstellen und nachvollziehen.

Selbst wenn Sie nicht die Absicht haben, eigene Pakete zu erstellen, finden Sie in diesem Kapitel hilfreiches Hintergrundwissen zu Portage.

15.1 Ein einfacher Ebuild

Wir werden uns hier mit der Installation eines Python-Paketes beschäftigen. Warum das besonders einfach ist, werden wir gleich sehen. Wir bleiben im Web-Bereich und schreiben einen Ebuild für `web.py`, das Web-Framework von Aaron Swartz.[1]

Das Paket ist schon im Portage-Baum verfügbar, aber auch nur, weil dieses Buch geschrieben wurde und es sinnvoll ist, den Ebuild gleich allgemein zur Verfügung zu stellen. Die einzelnen Schritte lassen sich trotzdem wie angegeben durchführen. Schauen wir uns also an, wie der Ebuild erstellt wurde.

[1] `http://webpy.org`

15.1.1 Den Ebuild schreiben

Zunächst brauchen wir einen Ort, an dem wir unseren neuen Ebuild bearbeiten, mit anderen Worten: unser erstes eigenes Overlay:

```
gentoo ~ # mkdir -p /usr/portage/local/overlay
```

Dieses fügen wir zu unseren definierten Overlays in `/etc/make.conf` hinzu:

```
PORTDIR_OVERLAY="/usr/portage/local/overlay"
```

Das Overlay ist natürlich noch leer und enthält keine Kategorien. Für unseren neuen Ebuild müssen wir uns also erst einmal für eine Kategorie entscheiden. Dabei orientieren wir uns an den bereits unter `/usr/portage` vorhandenen.

Für ein neues, eher entwicklerorientiertes Python-Paket kommt für `web.py` am ehesten `dev-python` in Frage. Für ein Web-Framework wären zwar auch `net-www` oder `www-misc` vorstellbar, allerdings liegt das vergleichbare `django`-Paket auch unter `dev-python`. Man sollte sich also an bereits integrierten und von der Funktionalität her vergleichbaren Paketen orientieren. Also erstellen wir `dev-python` als erste Kategorie unseres Overlays:

```
gentoo ~ # mkdir -p /usr/portage/local/overlay/dev-python/webpy
```

Wir erzeugen hier gleich noch das Verzeichnis für unseren ersten Ebuild und nennen das Paket `webpy`. Den Punkt von `web.py` lassen wir weg, denn Sonderzeichen gehören in den seltensten Fällen in Paketnamen. Fehlt noch die Angabe der Versionsnummer: `web.py` wurde als Version 0.22 herausgegeben und entsprechend heißt unser erster Ebuild `webpy-0.22.ebuild`. Damit ist es an der Zeit, den Ebuild selbst zu erstellen:

```
gentoo ~ # nano \
> /usr/portage/local/overlay/dev-python/webpy/webpy-0.22.ebuild
```

Was gehört nun in den Ebuild hinein? Zunächst der allgemeine Kopfbereich, der standardmäßig bei Gentoo verwendet wird. Wer nicht vorhat, den Ebuild im Portage-Baum zu sehen, kann natürlich darauf verzichten.

```
# Copyright 1999-2008 Gentoo Foundation
# Distributed under the terms of the GNU General Public License v2
# $Header:$
```

Ein Ebuild verwendet Shell-Syntax. Viele Hintergrundprozesse, die für den Installationsprozess wichtig sind, haben die Gentoo-Entwickler aber schon

in einfach zu verwendende Funktionen gekapselt, so dass wir uns im Ebuild auf die spezifischen Eigenschaften der zu installierenden Software konzentrieren können. Damit reduziert sich der Aufwand erheblich. Für kompliziertere Fälle steht aber dennoch die vollständige Funktionalität der Kommandozeile zur Verfügung.

Jedes Paket verlangt einige Standard-Informationen. Damit Außenstehende wissen, wozu das Paket überhaupt dient, ergänzen wir zunächst eine knappe Beschreibung in der Variablen `DESCRIPTION`.

```
DESCRIPTION="A small and simple web framework for python"
```

Der Hinweis auf die Homepage des Paketes kann Nutzern sicher ebenfalls weiterhelfen:

```
HOMEPAGE="http://www.webpy.org"
```

Dann ist natürlich der Source-Code verfügbar zu machen, und zwar in Form einer URL, damit `emerge` weiß, wo es das Quellpaket herunterladen kann. Den Link definieren wir in der Variablen `SRC_URI`. In unserem Beispiel lautet er: `http://www.webpy.org/static/web.py-0.22.tar.gz`.

In weiser Vorausschau wollen wir berücksichtigen, dass irgendwann wohl `web.py-0.23` oder `web.py-0.3` veröffentlicht und das Quellpaket dann unter `http://www.webpy.org/static/web.py-0.23.tar.gz` bzw. `web.py-0.3.tar.gz` abrufbar sein wird. Die Version unseres Paketes ist schon im Namen des Ebuilds enthalten, und Portage stellt diese Versionsnummer innerhalb eines Ebuilds in der Variablen `${PV}` zur Verfügung.

Wir können hier also die `0.22` im Quell-Link einfach durch `${PV}` ersetzen und werden so später den Ebuild vermutlich nur zu `webpy-0.23.ebuild` oder `webpy-0.3.ebuild` umbenennen müssen, und er wird trotzdem noch funktionieren.

```
SRC_URI="http://www.webpy.org/static/web.py-${PV}.tar.gz"
```

Nun fehlen noch einige Informationen, die in jeden Ebuild gehören, z. B. die Lizenz des Paketes, die die Variable `LICENSE` enthält. Entpackt man das Quellpaket, findet sich dort eine Datei `PKG-INFO`, in der `License: Public domain` angegeben ist. Bei Gentoo finden sich alle bisher bekannten Lizenzen unter `/usr/portage/licenses`.

Nur die Dateinamen der dort vorhandenen Textdateien dürfen wir innerhalb der Variable `LICENSE` angeben. Schaut man sich die Liste der innerhalb dieses Verzeichnisses vorhandenen Lizenzen an, so findet sich auch `public-domain` wieder. Entsprechend lautet die Angabe für `web.py`:

```
LICENSE="public-domain"
```

Das webpy-Paket soll nur einmal in unserem System installiert werden und normale Updates erfahren. Mit anderen Worten: Es verwendet keine Slots (siehe Kapitel 10.4.1) für die Installation, und emerge installiert es grundsätzlich in den Slot 0. Das ist eine Angabe, die in jeden Ebuild hinein gehört:

```
SLOT="0"
```

Zudem sollten wir noch angeben, auf welchen Maschinen das Paket sicher funktioniert, also die verfügbaren Keywords festlegen. Unter der Annahme, dass wir diesen Ebuild auf einer x86-Maschine entwickeln, fügen wir das entsprechende Keyword mit einer Tilde zu der Variablen KEYWORDS hinzu. Die Tilde markiert den Ebuild als instabil, was für einen neuen Ebuild zwingend erforderlich ist.

```
KEYWORDS="~x86"
```

Testet man den Ebuild auf mehreren Maschinen mit unterschiedlichen Architekturen, kann man mehrere Keywords durch Leerzeichen getrennt angeben.

Nun stellt sich die Frage nach eventuellen USE-Flags. Diese würden wir mit der Variablen IUSE angeben. Allerdings bietet web.py keine optionalen Eigenschaften, die wir bei der Installation aktivieren oder deaktivieren könnten, so dass wir auf USE-Flags verzichten und IUSE auf einen leeren Wert setzen:

```
IUSE=""
```

Grundsätzlich lassen sich USE-Flags in einer durch Leerzeichen getrennten Liste angeben.

Damit bleiben noch zwei Variablen, die für einen Ebuild zwingend erforderlich sind: DEPEND und RDEPEND. DEPEND gibt die Abhängigkeiten zur Installationszeit an, RDEPEND die Abhängigkeiten zur Laufzeit.

In Kapitel 4.2.2 haben wir ganz allgemein von Abhängigkeiten gesprochen und keine Unterscheidung zwischen verschiedenen Formen der Abhängigkeit getroffen. Für den Benutzer ist eine solche Unterscheidung im Normalfall auch nicht relevant, da Portage die Abhängigkeiten automatisch auflöst und nichts über die Art der Abhängigkeit mitteilt. Hinter den Kulissen, also aus der Sicht des Ebuilds ist diese Unterscheidung aber zwingend notwendig.

Unterschieden wird also zwischen *Build Time Dependencies* und *Run Time Dependencies*. Build Time Dependencies sind solche, die für das Kompilieren bzw. die Installation eines Paketes bestehen. So brauchen wir z. B. für ein in C geschriebenes Programm einen Compiler (im Normalfall gcc), damit wir es kompilieren und installieren können.

Nach der Installation können wir das Programm dann aber jederzeit aufrufen, ohne dass wir `gcc` dazu benötigen. Der Compiler ist also eine klare Build Time Dependency.

Run Time Dependencies umfassen alle Elemente, die im System installiert sein müssen, damit wir das Programm *ausführen* können. Das gilt z. B. für alle Bibliotheken, die ein Programm verwendet. Haben wir den Apache-Server z. B. mit SSL-Unterstützung kompiliert, muss die `openssl`-Bibliothek zur Laufzeit im System verfügbar sein. Das ist mit dem Beispiel vergleichbar, das wir in 10.4.4 konstruierten haben, um `revdep-rebuild` genauer zu erläutern.

Die beiden Arten der Abhängigkeit schließen sich übrigens nicht gegenseitig aus. Im Gegenteil gelten die meisten Abhängigkeiten sowohl für die Installationsphase als auch die Laufzeit. Die SSL-Bibliothek für den Apache muss eben auch schon beim Kompilieren des Apache vorhanden sein. Andernfalls lässt sich der Server nicht erfolgreich zusammenbauen.

Zurück zu `web.py`: Welche Software wird also benötigt, um `web.py` zu installieren? Wie die meisten Python-Pakete besitzt auch `web.py` ein Installationsskript `setup.py`, das auf dem `distutils`-Modul von Python aufbaut. Diese Kombination benötigt keine Software außer Python selbst für die Installation. Also legen wir die Abhängigkeiten folgendermaßen fest:

```
DEPEND="dev-lang/python"
```

Um `web.py` zu verwenden, brauchen wir natürlich auch wieder Python als Grundlage. Leider geht aus den Installationsanweisungen des Paketes `web.py` nicht eindeutig hervor, welche Python-Version mindestens benötigt wird. Wir sind an dieser Stelle einmal vorsichtig und nehmen an, dass es `python-2.3` voraussetzt. Sollten Nutzer später den Betrieb unter 2.1 oder 2.2 wünschen und zeigen können, dass es funktioniert, lassen sich die Abhängigkeiten auch nachträglich korrigieren. Um uns die Definition der Abhängigkeiten zu vereinfachen, geben wir auch für die Installation die Version 2.3 als Minimum an. Die Variablen sehen dann so aus:

```
DEPEND=">=dev-lang/python-2.3"
RDEPEND="${DEPEND}"
```

Damit liegen alle notwendigen Grundinformationen für einen Ebuild vor und wir können uns nun der eigentlichen Installation zuwenden. Bei der Installation gibt es drei Hauptphasen, die `emerge` durchläuft:

- Entpacken des Quellarchivs
- Kompilieren der Software
- Installation des Paketes

Jede dieser Phasen korrespondiert mit einer Funktion im Ebuild:

- `src_unpack`
- `src_compile`
- `src_install`

`src_unpack` ist meist gar nicht anzugeben, da Portage die meisten Standard-Archivformate problemlos entpackt. Die Funktion kann dann einfach fehlen. Bleiben das Kompilieren und das Installieren.

Ein Python-Paket, das `distutils` und das zugehörige Skript `setup.py` für die Installation verwendet, lässt sich im Normalfall mit folgender Kombination installieren:

```
gentoo ~ # python setup.py build
gentoo ~ # python setup.py install
```

`build` leitet das Kompilieren ein, während `install` sich um die Installation kümmert. Wir könnten also unseren Ebuild jetzt um folgende zwei Funktionen erweitern:

```
src_compile() {
        python setup.py build
}

src_install() {
        python setup.py install
}
```

Abgesehen davon, dass wir die Funktionen hier zu sehr vereinfacht haben, ist es wenig sinnvoll, in jedem Ebuild für ein Python-Modul diese oder ähnliche Zeilen anzugeben, da der `distutils`-Mechanismus unter Python-Paketen weit verbreitet ist. Wenn mehrere Ebuilds nach einem vereinheitlichten Schema installieren, findet sich in der Regel eine sogenannte *Eclass*, die den Installationsprozess für diese Pakete zentral definiert, so dass die eigentliche Paketdefinition nur noch ein Minimum an Aufwand kostet.

Die Grunddefinitionen einer Eclass bindet man im Ebuild über den Befehl `inherit` ein. Und für `distutils`-Pakete gibt es die passende Eclass unter `/usr/portage/eclasses/distutils.eclass`. Diese liefert die notwendigen Definitionen für `src_compile` und `src_install`.

In unserem Ebuild fügen wir also nur die folgende Zeile ein und können darauf verzichten, `src_compile` und `src_install` separat zu definieren:

```
inherit distutils
```

Eigentlich sind wir damit fertig. Das Einbinden der `distutils`-Eclass gibt alle Aktionen für das Entpacken, Erstellen und Installieren des Paketes vor, und da wir hier ein Standard-Paket vor uns haben, sind keine weiteren Modifikationen notwendig.

Nur eine Kleinigkeit fehlt noch: Wir haben uns anfangs dazu entschlossen, das Paket selbst nicht `web.py`, sondern `webpy` zu nennen, um keine Sonderzeichen im Paketnamen zu haben. Nun geht die `distutils`-Eclass aber davon aus, dass unser Quellpaket im Namen der Paketbezeichnung entspricht. Zwar ist Portage in der Lage, anhand des Links zum Quellarchiv zu erkennen, dass unser Quellpaket eben `web.py-0.22.tar.gz` heißt, und würde entsprechend auch dieses Archiv auspacken. Allerdings erwartet Portage im folgenden Schritt, dass innerhalb des Archivs ein Quellverzeichnis mit dem Namen `webpy-0.22`, eben unserem Paketnamen, liegt. Dies ist nicht der Fall, denn das Quellverzeichnis wurde von den Entwicklern einfach nur `webpy` genannt.

Den Namen des entpackten Verzeichnisses legen wir über die Variable `${S}` fest. Der Standardwert ist `${WORKDIR}/${PN}-${PV}`. `${WORKDIR}` legt das Arbeitsverzeichnis fest und verweist auf ein temporäres Verzeichnis unter `/var/tmp/portage`. `${PV}` steht, wie bereits erwähnt, für die Paketversion und `${PN}` ist der Paketname. Diesen leitet Portage bei der Installation automatisch vom Ebuild-Namen ab, und er wird hier auf `webpy` gesetzt. Damit korrigieren wir `${S}`, indem wir einfach die Paketversion entfernen:

```
S="${WORKDIR}/${PN}"
```

Damit haben wir unseren ersten Ebuild vollständig definiert, und er sollte nun wie folgt aussehen:

```
# Copyright 1999-2008 Gentoo Foundation
# Distributed under the terms of the GNU General Public License v2
# $Header:$

inherit distutils

DESCRIPTION="A small and simple web framework for python"
HOMEPAGE="http://www.webpy.org"
SRC_URI="http://www.webpy.org/static/web.py-${PV}.tar.gz"

LICENSE="public-domain"
SLOT="0"
KEYWORDS="~x86"
IUSE=""

DEPEND=">=dev-lang/python-2.3"
RDEPEND="${DEPEND}"

S="${WORKDIR}/${PN}"
```

Die Reihenfolge unterscheidet sich etwas von der Reihenfolge der Erläuterung, da am Anfang des Ebuilds die `inherit`-Direktive stehen sollte.

15.1.2 Den Ebuild zu einem Paket umwandeln

Noch sind wir aber nicht so weit, dass wir über `emerge webpy` die Installation anstoßen könnten. Zu einem vollständigen Paket gehören noch die Prüfsummen, die es Portage erlauben, alle Elemente eines Paketes auf den korrekten Inhalt zu überprüfen. Der Entwickler signiert damit praktisch die Einzelteile des Paketes, und der Nutzer kann somit davon ausgehen, dass der Ebuild genau die vom Entwickler vorgegebenen Aktionen durchführt – wenn alle Prüfsummen übereinstimmen. Andernfalls wird Portage den Installationsvorgang abbrechen.

Das Signieren des Paketes erstellt einen sogenannten *Digest* der Paket-Einzelteile. Dafür brauchen wir das Kommando `ebuild` aus dem `portage`-Paket:

```
gentoo ~ # cd /usr/portage/local/overlay/dev-python/webpy
gentoo webpy # ebuild webpy-0.22.ebuild digest
```

Das Tool lädt nun das Quellpaket in `/usr/portage/distfiles` und endet dann mit folgender Nachricht:

```
>>> Creating Manifest for /usr/portage/local/overlay/dev-python/webpy
```

Der Inhalt unseres Paket-Verzeichnisses hat sich etwas erweitert:

```
gentoo webpy # ls -la
insgesamt 20
drwxr-xr-x 3 root root 4096  1. Feb 13:39 .
drwxr-xr-x 3 root root 4096  1. Feb 13:37 ..
drwxr-xr-x 2 root root 4096  1. Feb 13:39 files
-rw-r--r-- 1 root root  855  1. Feb 13:39 Manifest
-rw-r--r-- 1 root root  440  1. Feb 13:38 webpy-0.22.ebuild
```

Im `files`-Unterverzeichnis befindet sich nur der Digest des Quellarchivs:

```
gentoo webpy # cat files/digest-webpy-0.22
MD5 2e7b5b6759a507d6480fa18d6e87a636 web.py-0.22.tar.gz 52697
RMD160 ae00772c79928722a3627fdb958f1b7378917f3b web.py-0.22.tar.gz 52697
SHA256 f2359c8a711660c4d7b910d22c770058dc548c3bab7a8fec0cba4f99758a379c
web.py-0.22.tar.gz 52697
```

Portage verwendet hier drei verschiedene Prüfsummen mit unterschiedlichem Sicherheitsniveau. Ähnliche Informationen finden sich im neu erstellten `Manifest` des Paketes wieder.

```
gentoo webpy # cat Manifest
DIST web.py-0.22.tar.gz 52697 RMD160 ae00772c79928722a3627fdb958f1b73789
17f3b SHA1 ee8910ee2992f6212780f24c611fb479c8cee206 SHA256 f2359c8a71166
0c4d7b910d22c770058dc548c3bab7a8fec0cba4f99758a379c
EBUILD webpy-0.22.ebuild 440 RMD160 0a5baf74c86970d55546954a2e01fcb76a66
b765 SHA1 c7b400b96171f6764e066e1f50f0e0d3fa502a26 SHA256 1a380c3c839cf7
53e1bcf639b5e470a7188739c050440f63498f522828890544
MD5 ced82b80a5d5554b59e39f6aec31e4b5 webpy-0.22.ebuild 440
RMD160 0a5baf74c86970d55546954a2e01fcb76a66b765 webpy-0.22.ebuild 440
SHA256 1a380c3c839cf753e1bcf639b5e470a7188739c050440f63498f522828890544
webpy-0.22.ebuild 440
MD5 ba3bc8cd592f617d9f30654c76521c2e files/digest-webpy-0.22 232
RMD160 d44173553e140341e64acfcfff854ec27d017195 files/digest-webpy-0.22
232
SHA256 6e66c9d880d00273931721c07f5abd7f02b508d5cf02ce9547f0147401b3f896
files/digest-webpy-0.22 232
```

Hier sind nun alle Dateien des Paketes aufgeführt, und auf dieser Basis kann Portage vollständig verifizieren, dass der Ebuild beim Nutzer die „originalen“ Quellen verwendet. Dies ist wichtig, um das Einschleusen von Schadcode zu verhindern.

Nun sind wir so weit, den Ebuild erstmals zu installieren:

```
gentoo webpy # emerge -pv webpy

These are the packages that would be merged, in order:

Calculating dependencies
!!! All ebuilds that could satisfy "webpy" have been masked.
!!! One of the following masked packages is required to complete your re
quest:
- dev-python/webpy-0.22 (masked by: ~x86 keyword)

For more information, see MASKED PACKAGES section in the emerge man page
 or
refer to the Gentoo Handbook.
```

Portage verweigert die Installation, da wir den Ebuild mit `~x86` als instabil markiert haben. Natürlich vertrauen wir unserem eigenen Ebuild und akzeptieren das instabile Keyword. Dafür hängen wir das benötigte `dev-python/webpy ~x86` an die Datei `/etc/portage/package.keywords` an, bevor wir es noch einmal probieren:

```
gentoo webpy # flagedit dev-python/webpy -- ~x86
gentoo webpy # emerge -pv webpy

These are the packages that would be merged, in order:

Calculating dependencies... done!
```

```
[ebuild  N    ] dev-python/webpy-0.22  0 kB [1]

Total size of downloads: 0 kB
Portage overlays:
 [1] /usr/portage/local/overlay
gentoo webpy # cd ~
gentoo ~ #
```

Portage zeigt nun korrekt an, dass wir die neueste Version 0.22 installieren würden, das Quellarchiv nicht mehr herunterladen müssen und der Ebuild aus einem Overlay, eben `/usr/portage/local/overlay`, stammt.

15.2 Der ebuild-Befehl

Wir haben den Befehl `ebuild` bereits kennen gelernt und damit den Digest für unseren Ebuild erstellt. Das `ebuild`-Kommando zählt generell zu den Entwickler-Werkzeugen und eignet sich recht gut, um die eigentliche Struktur eines Ebuilds zu verdeutlichen.

Wir haben im letzten Abschnitt zwar einen Ebuild erstellt, aber im Grunde nur einige allgemeine Informationen über unser Paket in Variablen festgehalten. Aus diesen Informationen ist kaum zu ersehen, wie das Paket eigentlich heruntergeladen, ausgepackt, kompiliert und installiert wird.

Diese Einzelschritte führt `emerge` bei einer Paket-Installation hintereinander aus. Sie lassen sich über den `ebuild`-Befehl in einzelne Teile herunterbrechen.

Der erste Schritt einer Installation über `emerge` besteht aus dem Herunterladen des Quellarchivs und dem Überprüfen der Digests. Diese Aktion können wir separat mit Hilfe von `ebuild` und dem Kommando `fetch` durchführen:

```
gentoo ~ # cd /usr/portage/local/overlay/dev-python/webpy
gentoo webpy # ebuild webpy-0.22.ebuild fetch
 * web.py-0.22.tar.gz MD5 ;-) ...                                    [ ok ]
 * web.py-0.22.tar.gz RMD160 ;-) ...                                 [ ok ]
 * web.py-0.22.tar.gz SHA1 ;-) ...                                   [ ok ]
 * web.py-0.22.tar.gz SHA256 ;-) ...                                 [ ok ]
 * web.py-0.22.tar.gz size ;-) ...                                   [ ok ]
 * checking ebuild checksums ;-) ...                                 [ ok ]
 * checking auxfile checksums ;-) ...                                [ ok ]
 * checking miscfile checksums ;-) ...                               [ ok ]
 * checking web.py-0.22.tar.gz ;-) ...                               [ ok ]
```

Wir sehen hier die bekannten ersten Zeilen des normalen `emerge`-Prozesses. Es fehlt nur das Herunterladen des Quellcodes, da wir das Quellarchiv ja schon nach `/usr/portage/distfiles` heruntergeladen haben. Wir sehen,

dass alle Prüfsummen dieses Archivs als korrekt erkannt werden, was zu erwarten war, da wir auf Basis des gleichen Quellarchivs den Digest überhaupt erst erstellt haben. Dieser Check ergibt natürlich erst auf der Nutzer-Seite Sinn. Auch die Prüfsummen des Ebuilds und anderer Dateien überprüft `ebuild` hier erfolgreich.

Wurde das Paket erfolgreich heruntergeladen und überprüft, ist es an der Zeit, die Quellen zu extrahieren und die eigentlichen Vorbereitungen für die Installation zu treffen. Dieser Prozess lässt sich wieder über `ebuild` abbilden, diesmal mit dem Kommando `unpack`:

```
gentoo webpy # ebuild webpy-0.22.ebuild unpack
...
 * checking web.py-0.22.tar.gz ;-) ...                        [ ok ]
>>> Unpacking source...
>>> Unpacking web.py-0.22.tar.gz to /var/tmp/portage/webpy-0.22/work
>>> Source unpacked.
```

Diese Aktion prüft die Digests nochmals und packt das Archiv dann aus. Portage gibt auch den Ort an, wo es die Quellen temporär speichert: `/var/tmp/portage/webpy-0.22/work`. Dieser Pfad entspricht der Variable `${WORKDIR}`.

Schauen wir uns den Inhalt dieses temporären Verzeichnisses an, finden wir dort das einzelne Verzeichnis, das im Quellarchiv enthalten war:

```
gentoo webpy # ls -la  /var/tmp/portage/dev-python/webpy-0.22/work
insgesamt 12
drwx------ 3 root    root    4096  1. Feb 13:46 .
drwxrwxr-x 6 portage portage 4096  1. Feb 13:46 ..
drwxr-xr-x 4 root    root    4096 23. Aug 07:04 webpy
```

Sind die Quellen ausgepackt, gilt es, das Paket zu kompilieren und damit installierbare Dateien zu erstellen. Hier hilft die Aktion `compile`:

```
gentoo webpy # ebuild webpy-0.22.ebuild compile
...
>>> WORKDIR is up-to-date, keeping...
>>> Compiling source in /var/tmp/portage/webpy-0.22/work/web.py-0.22 ...
running build
running build_py
creating build
creating build/lib
creating build/lib/web
copying web/wsgi.py -> build/lib/web
copying web/cheetah.py -> build/lib/web
copying web/db.py -> build/lib/web
copying web/template.py -> build/lib/web
copying web/form.py -> build/lib/web
```

```
copying web/net.py -> build/lib/web
copying web/request.py -> build/lib/web
copying web/httpserver.py -> build/lib/web
copying web/debugerror.py -> build/lib/web
copying web/http.py -> build/lib/web
copying web/__init__.py -> build/lib/web
copying web/webapi.py -> build/lib/web
copying web/utils.py -> build/lib/web
>>> Source compiled.
```

Wieder überprüft `ebuild` die Digests, stellt fest, ob das `${WORKDIR}` noch aktuell ist, und kompiliert dann die Quellen. Für ein `distutils`-Paket führt Portage den Befehl `python setup.py build` aus. Dieser stammt erwartungsgemäß aus der Eclass `distutils`.

Eine Aktion, die an dieser Stelle stehen könnte, jedoch von unserem derzeitigen Ebuild nicht unterstützt wird, ist das automatische Testen des Paketes. Manche Software bringt Test-Skripte oder so genannte Unit-Tests mit, die helfen, die Funktionalität der Software automatisiert zu überprüfen. Python-Module liefern vielfach in der Dokumentation versteckte Tests (*Doctests*) mit, und auch innerhalb von `web.py` finden sich einige. Vor allem auf den weniger verbreiteten Architekturen finden diese Tests häufig noch überraschende Inkompatibilitäten.

Wir wollen unseren Ebuild also um diese Tests erweitern. Tests deckt die `distutils`-Eclass nicht automatisch ab, und so brauchen wir unsere erste wirkliche Shell-Funktion, die `src_test` heißen muss:

```
src_test() {
        # Diese Dateien enthalten automatische doctests
        TESTS="db template net http utils"

        # Wir wechseln an dieser Stelle in das Arbeitsverzeichnis
        cd ${S}

        # Nun durchlaufen wir jede der oben genannten Dateien die
        # doctests enthalten
        for TEST in ${TESTS}
        do
                # Für jeden der zu testenden Dateien rufen wir die
                # entsprechende Datei mit dem Python-Interpreter auf
                # Das führt zum Durchlaufen der doctests.
                # Sollte einer der doctests scheitern, wird der Ebuild
                # abbrechen.
                ${python} web/${TEST}.py || die "Doctest failed!"
        done
}
```

Die Test-Funktion ist recht kurz gehalten und wertet nur die unter `${TESTS}` angegebenen Dateien im Python-Interpreter aus. Das bewirkt für die jewei-

lige Datei das Ausführen der *Doctests*. Sollte es dabei zu Fehlern kommen, bricht `emerge` den Vorgang ab.

Da wir jetzt den Ebuild modifiziert haben, stimmt der Digest nicht mehr, und wir müssen ihn neu generieren. Erst danach können wir den `ebuild`-Befehl mit der Aktion `test` aufrufen. Dabei gilt es noch eine Besonderheit zu beachten: Normalerweise wird Portage die `src_test`-Funktion nicht in den `emerge`-Prozess (oder eben auch in die `ebuild webpy-0.22.ebuild test`-Aktion) einbeziehen. Das liegt daran, dass die Tests einiger Pakete dermaßen viel Zeit kosten (z. B. `glibc`), dass dieses Feature für den durchschnittlichen Nutzer nicht sinnvoll ist. Man muss es also explizit in der `FEATURES`-Option von `/etc/make.conf` aktivieren (siehe Kapitel 6.3.3 ab Seite 156). Alternativ lässt es sich auch einmalig über die Kommandozeile aktivieren:

```
gentoo webpy # ebuild webpy-0.22.ebuild digest
gentoo webpy # FEATURES="test" ebuild webpy-0.22.ebuild test
...
>>> Source compiled.
```

Im vorliegenden Fall sollte nichts passieren und die Doctests einwandfrei funktionieren. Nach `Source compiled` erfolgt in diesem Fall kein weiterer Output.

Kommen wir nach diesem Einschub zurück zum normalen Verlauf einer Installation: Nachdem `ebuild` die Komponenten für die Installation vorbereitet hat, müssen die Dateien installiert werden. Portage wird dabei die Installation in ein temporäres Verzeichnis vornehmen und erst dann in das eigentliche System übertragen. Damit kann Portage genau verfolgen, welche Dateien es installiert, und es so ermöglichen, die Software später weitgehend rückstandsfrei zu entfernen.

```
gentoo webpy # ebuild webpy-0.22.ebuild install
...
copying build/lib/web/__init__.py -> /var/tmp/portage/webpy-0.22/image/u
sr/lib/python2.4/site-packages/web
copying build/lib/web/webapi.py -> /var/tmp/portage/webpy-0.22/image/usr
/lib/python2.4/site-packages/web
copying build/lib/web/utils.py -> /var/tmp/portage/webpy-0.22/image/usr/
lib/python2.4/site-packages/web
>>> Completed installing webpy-0.22 into /var/tmp/portage/webpy-0.22/ima
ge/
```

Wie angegeben, erfolgt die Installation in das temporäre Verzeichnis `/var/tmp/portage/webpy-0.22/image/`.

Aus diesem erfolgt die eigentliche und abschließende Installation über die Aktion `qmerge`. Dabei wird Portage die installierten Dateien in der Datenbank unter `/var/db/pkg/dev-python/webpy-0.22/CONTENTS` festhalten.

```
gentoo webpy # ebuild webpy-0.22.ebuild qmerge
...
>>> Merging dev-python/webpy-0.22 to /
--- /usr/
--- /usr/lib/
--- /usr/lib/python2.4/
--- /usr/lib/python2.4/site-packages/
>>> /usr/lib/python2.4/site-packages/web/
>>> /usr/lib/python2.4/site-packages/web/wsgi.py
>>> /usr/lib/python2.4/site-packages/web/cheetah.py
>>> /usr/lib/python2.4/site-packages/web/db.py
>>> /usr/lib/python2.4/site-packages/web/template.py
>>> /usr/lib/python2.4/site-packages/web/form.py
>>> /usr/lib/python2.4/site-packages/web/net.py
>>> /usr/lib/python2.4/site-packages/web/request.py
>>> /usr/lib/python2.4/site-packages/web/httpserver.py
>>> /usr/lib/python2.4/site-packages/web/debugerror.py
>>> /usr/lib/python2.4/site-packages/web/http.py
>>> /usr/lib/python2.4/site-packages/web/__init__.py
>>> /usr/lib/python2.4/site-packages/web/webapi.py
>>> /usr/lib/python2.4/site-packages/web/utils.py
--- /usr/share/
--- /usr/share/doc/
>>> /usr/share/doc/webpy-0.22/
>>> /usr/share/doc/webpy-0.22/PKG-INFO.gz
 * Performing Python Module Cleanup .. ...
 * Cleaning orphaned Python bytecode from /usr/lib/python2.3/site-packag
es ..
 * Cleaning orphaned Python bytecode from /usr/lib/python2.4/site-packag
es ..                [ ok ]
>>> Regenerating /etc/ld.so.cache...
gentoo webpy # cd ~
gentoo ~ #
```

Damit sind wir einmal erfolgreich durch den Ablauf einer Installation gewandert und haben die zentralen Elemente des Prozesses kennen gelernt.

Kapitel

16 Tipps und Tricks

Wir nähern uns dem Ende unserer Reise durch das Gentoo-System und schließen nach der ausführlichen Darstellung zentraler Mechanismen und Werkzeuge mit einer kleinen Sammlung von Tipps und Tricks. Diese folgende Zusammenstellung erhebt natürlich keinen Anspruch auf Vollständigkeit, aber hält einige Empfehlungen bereit, die den Umgang mit Gentoo zusätzlich vereinfachen; sie sollen darum nicht unerwähnt bleiben.

16.1 Werkzeuge aus der Kategorie app-portage

Wie der Name der Kategorie verrät, finden wir hier spezielle Werkzeuge für das Paketmanagement. Manche dieser Pakete sind eher auf die Bedürfnisse von Entwicklern zugeschnitten, aber es finden sich hier auch einige Juwelen für jeden Gentoo-Nutzer.

Wir beschränken uns hier auf `mirrorselect` und `getdelta`; es sei jedoch empfohlen, die gesamte Kategorie einmal zu durchstöbern.

16.1.1 Spiegel-Server auswählen: app-portage/mirrorselect

`mirrorselect` ist ein Gentoo-spezifisches Werkzeug für die Auswahl des optimalen Mirror-Servers. Das Skript befindet sich im Portage-Baum unter `app-portage/mirrorselect` und ist folglich zu installieren über:

```
gentoo ~ # emerge -av app-portage/mirrorselect

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-portage/mirrorselect-1.2  0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
```

Es ist ursprünglich für die Auswahl der Server mit den Quellpaketen gedacht, aber wir können es auch für die Selektion des Rsync-Servers, der den Portage-Baum liefert, nutzen.

Interaktiver Modus

`mirrorselect` kann automatisch suchen, erlaubt aber auch Eingaben des Benutzers. Für den interaktiven Modus startet man `mirrorselect` mit der Option `--interactive` (bzw. `-i`) und landet in einem grafischen Menü mit den verfügbaren Servern.

Nach Beenden des Programms finden sich die neuen Server in der Datei `/etc/make.conf` im Parameter `GENTOO_MIRRORS`; die ursprüngliche, d. h. unveränderte Datei wurde zuvor nach `/etc/make.conf.backup` verschoben.

Wer bei den Veränderungen noch Zweifel hegt und `mirrorselect` nur testen möchte, sollte das Programm mit der Option `--output` (bzw. `-o`) starten. In diesem Fall gibt `mirrorselect` bei Beenden nur den neuen Wert von `GENTOO_MIRRORS` auf der Kommandozeile aus. Ist man mit dem neuen Wert zufrieden, kann man ihn manuell in `/etc/make.conf` eintragen oder das Tool noch einmal ohne die Option `-o` laufen lassen.

Da es ohnehin nur ein halbes Dutzend zentrale Rsync-Mirror gibt, die die Lastverteilung auf untergeordnete Rsync-Server automatisch übernehmen, ist die Selektion des Rsync-Mirrors nur interaktiv sinnvoll.

Für das entsprechende Menü fügt man zu der Option `-i` einfach noch `--rsync` (bzw. `-r`) hinzu:

```
gentoo ~ # mirrorselect -i -r
```

Automatischer Modus

Dies ist sicher der angenehmste und effizienteste Weg, die eigenen Mirror-Server zu bestimmen. Im automatischen Modus lädt `mirrorselect` von jedem verfügbaren Server ein kurzes Datensegment herunter und ermittelt anhand der Reaktionszeit und Übertragungsgeschwindigkeit, welcher aktuell der günstigste ist.

Wichtigster Parameter im automatischen Modus ist die Anzahl alternativer Server, die `mirrorselect` auswählen soll. Diesen Wert können wir mit der Option `--servers` (bzw. `-s`) spezifizieren.

Auch im automatischen Modus verändert `mirrorselect` die Datei `/etc/make.conf` unmittelbar und legt ein Backup der Ursprungsdatei als `/etc/make.conf.backup` ab. Dieses Verhalten lässt sich ebenso wie im interaktiven Modus mit `-o` unterdrücken.

```
gentoo ~ # mirrorselect -s3 -o
* Downloading a list of mirrors... Got 223 mirrors.
* Stripping hosts that only support ipv6... Removed 8 of 223
* Using netselect to choose the top 3 mirrors...Done.

GENTOO_MIRRORS="
http://pandemonium.tiscali.de/pub/gentoo/
ftp://ftp.snt.utwente.nl/pub/os/linux/gentoo
ftp://213.186.33.37/gentoo-distfiles/"
```

`mirrorselect` wählt, wie in der Ausgabe zu sehen, sowohl HTTP- als auch FTP-Server aus. Wer sich auf eine Variante beschränken möchte kann dies mit der Option `--ftp` (bzw. `-F`) oder eben `--http` (bzw. `-H`) veranlassen.

Wer bei der massiven Abfrage von Servern Probleme mit dem Router bekommt, weil dieser eine hohe Zahl gleichzeitiger Anfragen nicht akzeptiert, kann die Zahl zeitgleicher Verbindungen über die Option `--blocksize` (bzw. `-b`) regulieren.

```
gentoo ~ # mirrorselect -s3 -b10 -H -o
* Downloading a list of mirrors... Got 223 mirrors.
* Limiting test to http hosts. 123 of 223 removed.
* Stripping hosts that only support ipv6... Removed 2 of 100
Using netselect to choose the top3 hosts, in blocks of 10. 10 of 10 bloc
ks complete.

GENTOO_MIRRORS="
http://ftp.gentoo.or.kr/
http://ftp.twaren.net/Linux/Gentoo/
http://ftp.isu.edu.tw/pub/Linux/Gentoo"
```

Zu guter Letzt lässt sich die automatische Server-Auswahl ein wenig optimieren, indem man höhere Datenmengen von den potentiellen Servern

herunterlädt. Mit der Option `--deep` (bzw. `-D`) lädt `mirrorselect` pro Server als Test ein 100-Kilobyte-Segment herunter. Damit wird die Messung und die resultierende Selektion deutlich präziser. Gleichzeitig schwillt der Datentransfer bei zweihundert zu testenden Servern auch auf 20 MB an, so dass sich diese Tests nur mit einer geeigneten Datenleitung empfehlen.

Damit diese Tests bei Server-Fehlern nicht zu lange dauern, kann man den Timeout-Wert für die Verbindungen mit der Option `--timeout` (bzw. `-t`) regulieren.

Allgemein gültig sind auch die Optionen `--quiet` (bzw. `-q`) und `--debug` (bzw. `-d`), die den Output von `mirrorselect` reduzieren bzw. erhöhen.

16.1.2 Downloads optimieren: app-portage/getdelta

Wie schon mehrfach betont, ist eine Netzwerkverbindung für den Betrieb eines Gentoo-Systems nahezu unerlässlich. Angesichts der heutigen DSL-Durchsatzraten haben wir darauf verzichtet, die Menge herunterzuladender Daten zu berücksichtigen. Nur einmal, im Zusammenhang mit `emerge-delta-webrsync` (siehe Seite 209), ging es um eine Reduktion der Download-Menge.

Aber es ist natürlich nicht jeder mit einem volumenmäßig unbeschränkten Breitbandzugang ausgestattet, und wir beschreiben einen einfachen Weg, die Download-Menge beim Aktualisieren mit dem Skript `getdelta.sh` wirkungsvoll zu beschränken.

Das zugehörige Paket `app-portage/getdelta` ist leider nicht stabil markiert, und auch einige der von ihm benötigten Pakete sind nur „instabil“ verfügbar. Wir bedienen uns also `flagedit`, um die Pakete trotzdem in unserem System zu installieren:

```
gentoo ~ # flagedit app-portage/deltup -- ~x86
gentoo ~ # flagedit dev-util/bdelta -- ~x86
gentoo ~ # flagedit =app-arch/bzip2-1.0.4 -- ~x86
gentoo ~ # flagedit app-portage/getdelta -- ~x86
gentoo ~ # emerge -av app-portage/getdelta

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] dev-util/bdelta-0.1.0  8 kB
[ebuild     U ] app-arch/bzip2-1.0.4 [1.0.3-r6] USE=''-static (-build%)"
 822 kB
[ebuild  N    ] app-portage/deltup-0.4.3_pre2-r1  664 kB
[ebuild  N    ] app-portage/getdelta-0.7.7  11 kB

Total: 4 packages (1 upgrade, 3 new), Size of downloads: 1,503 kB
```

```
Would you like to merge these packages? [Yes/No] Yes
...
```

Haben wir die Pakete installiert, ist die abschließende Konfiguration trivial: Wir ersetzen in der Datei `/etc/make.conf` den Inhalt der Variablen `FETCHCOMMAND` mit folgendem Wert:

```
FETCHCOMMAND="/usr/bin/getdelta.sh \${URI}"
```

Normalerweise enthält `FETCHCOMMAND` direkt den `wget`-Befehl, der eine ihm übergebene URL zu einem Quellarchiv herunterlädt. Wir ersetzen also dieses direkte Vorgehen mit einem Aufruf des `getdelta.sh`-Skriptes.

Dieses Skript geht an den Download des Quellarchivs nun etwas intelligenter heran als `wget`. So schaut `getdelta.sh` zunächst einmal in `/usr/portage/distfiles` nach, ob wir schon eine ältere Version des Quellarchivs besitzen und wird, sofern es eine solche findet, versuchen, nur ein Differenz-Archiv der vorhandenen und der benötigten Version herunterzuladen.

Veranschaulicht sei das am Paket `dev-libs/openssl`. Aktuell haben wir die Version `0.9.8d` installiert und damit auch das entsprechende Quellarchiv vorliegen:

```
gentoo ~ # ls -la /usr/portage/distfiles/openssl-*
-rw-rw-r-- 1 root portage 3294357 29. Jan 20:17 /usr/portage/distfiles/o
penssl-0.9.7l.tar.gz
-rw-rw-r-- 1 root portage 3315566 28. Jan 21:16 /usr/portage/distfiles/o
penssl-0.9.8d.tar.gz
```

Wir wollen nun die nächste Version des Pakets installieren und simulieren dies zunächst, indem wir `ACCEPT_KEYWORDS` setzen und `emerge` aufrufen:

```
gentoo ~ # ACCEPT_KEYWORDS="~x86" emerge -pv dev-libs/openssl

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild     U ] dev-libs/openssl-0.9.8e [0.9.8d] USE="zlib -bindist -ema
cs -sse2 -test" 3,264 kB
```

Ein Upgrade ist demnach verfügbar. Wir wollen nun mit der `--fetch`-Option nur das Quellarchiv herunterladen und schauen uns an, wie `emerge` vorgeht, wenn `getdelta.sh` als `FETCHCOMMAND` dient:

```
gentoo ~ # ACCEPT_KEYWORDS="~x86" emerge -f dev-libs/openssl

Calculating dependencies... done!
```

```
>>> Emerging (1 of 1) dev-libs/openssl-0.9.8e to /
>>> Downloading 'ftp://pandemonium.tiscali.de/pub/gentoo/distfiles/opens
sl-0.9.8e.tar.gz'

Searching for a previously downloaded file in /usr/portage/distfiles

We have the following candidates to choose from
openssl-0.9.7l.tar.gz
openssl-0.9.8d.tar.gz

The best of all is ... openssl-0.9.8d.tar.gz

Checking if this file is OK.

Trying to download openssl-0.9.8d.tar.gz-openssl-0.9.8e.tar.gz.dtu

...

GOT openssl-0.9.8d.tar.gz-openssl-0.9.8e.tar.gz.dtu

Successfully fetched the dtu-file - let's build openssl-0.9.8e.tar.gz...

openssl-0.9.8d.tar.gz -> openssl-0.9.8e.tar.gz: OK
cleaning up
This dtu-file saved 3 MB (98%) download size.
...
```

`getdelta.sh` gibt einen sehr genauen Überblick über seine Aktivitäten. So sucht es zuerst nach alten Quellarchiven in `/usr/portage/distfiles` (`Searching for a previously downloaded file in /usr/portage/distfiles`). Dabei identifiziert es `openssl-0.9.8d.tar.gz` als das Quellarchiv mit der höchsten Versionsnummer (`The best of all is ... openssl-0.9.8d.tar.gz`) und versucht, ein Differenz-Archiv mit Namen `openssl-0.9.8d.tar.gz-openssl-0.9.8e.tar.gz.dtu` zu laden (`Trying to download openssl-0.9.8d.tar.gz-openssl-0.9.8e.tar.gz.dtu`). Da das gelingt, fügt es das alte Archiv mit der Differenz zusammen und erstellt daraus das Zielarchiv `openssl-0.9.8e.tar.gz`.

Und damit sich abschließend der Nutzer auch über den Vorgang freuen kann, informiert `getdelta.sh`, dass 98 Prozent der Downloadmenge eingespart wurden (`This dtu-file saved 3 MB (98%) download size.`). Keine schlechte Leistung.

Ein Gentoo-System, dessen Netzwerkzugang auf ein Modem beschränkt ist, bleibt zwar trotz `getdelta.sh` eine grenzwertige Erfahrung, aber Nutzer mit Tarifen auf Basis der Download-Menge können sich damit über geringere Rechnungen freuen.

16.2 Werkzeuge aus der Kategorie app-admin

Diese Kategorie bietet allgemeinere Werkzeuge zur Administration eines Linux-Systems. Sie umfasst deutlich weniger Gentoo-spezifische Pakete, und wir wollen hier auch nur zwei herausgreifen: `app-admin/logrotate`, mit dem wir kurz das Management von Log-Files unter Gentoo beleuchten, und `app-admin/localepurge`, das interessante Eigenschaften von Portage veranschaulicht.

16.2.1 Logs aufräumen: app-admin/logrotate

Auf jedem System legen verschiedene Software-Pakete unter `/var/log` Log-Dateien an. Diese zeichnen wichtige System-Ereignisse, Fehler und andere Informationen auf, die für den Benutzer/Administrator wichtig sein könnten.

So legt z. B. der Apache ein eigenes Verzeichnis unter `/var/log/apache2` an und verzeichnet dort alle Zugriffe auf die angebotenen Webseiten. Gerade auf hoch frequentierten Servern können die entsprechenden Log-Dateien eine beträchtliche Größe erreichen. Darum ist es sinnvoll, hier in regelmäßigen Abständen aufzuräumen, d. h. Log-Dateien zu komprimieren und zu archivieren, was insbesondere bei Textdateien häufig hohe Platzersparnis bringt.

Natürlich gibt es hier ein Standard-Paket, das diese Aufgabe effizient erledigt: `app-admin/logrotate`.

```
gentoo ~ # emerge -av app-admin/logrotate

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] app-admin/logrotate-3.7.2  USE="(-selinux)" 0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
 * If you wish to have logrotate e-mail you updates, please
 * emerge mail-client/mailx and configure logrotate in
 * /etc/logrotate.conf appropriately
 *
 * Additionally, /etc/logrotate.conf may need to be modified
 * for your particular needs.  See man logrotate for details.
...
```

Zum Abschluss der Installation informiert das Paket, welche Konfiguration notwendig ist, um einen Bericht der `logrotate`-Aktivitäten an eine E-Mail-

Adresse zu versenden. Meist hat `logrotate` jedoch nicht viel zu berichten und wir verzichten auf eine entsprechende Konfiguration.

`logrotate` ist kein großes Paket und liefert außer dem eigentlichen Programm `/usr/sbin/logrotate` die Dokumentation und einige Konfigurationsdateien:

```
gentoo ~ # qlist app-admin/logrotate
/etc/cron.daily/logrotate.cron
/etc/logrotate.d/.keep_app-admin_logrotate-0
/etc/logrotate.conf
/usr/sbin/logrotate
/usr/share/doc/logrotate-3.7.2/logrotate.cron.gz
/usr/share/doc/logrotate-3.7.2/logrotate-default.gz
/usr/share/man/man8/logrotate.8.gz
```

Schauen wir uns kurz die Gentoo-Konfiguration an:

```
gentoo ~ # cat /etc/logrotate.conf
#
# Logrotate default configuration file for Gentoo Linux
#
# See "man logrotate" for details

# rotate log files weekly
weekly
#daily

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
compress

# packages can drop log rotation information into this directory
include /etc/logrotate.d

notifempty
nomail
noolddir

# no packages own lastlog or wtmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be also be configured here.
```

Auf diese Weise räumt `logrotate` die Log-Files wöchentlich (`weekly`) auf. Wer einen stark frequentierten Server betreibt, kann die Frequenz auf ein tägliches Aufräumen verkürzen: `weekly` auskommentieren und `daily` aktivieren.

`rotate 4` führt dazu, dass `logrotate` maximal vier alte Log-Archive aufbewahrt. Mit dem wöchentlichen Rhythmus behält man so die Log-Informationen eines Monats. Wer sich für das tägliche Aufräumen entscheidet, sollte die Zahl der archivierten Dateien entsprechend erhöhen.

`create` weist `logrotate` an, das alte Log-File nicht nur zu archivieren und zu verschieben, sondern wieder eine gleich benannte, aber leere Datei zu erstellen.

Mit der Option `compress` komprimiert `logrotate` die Archivdateien mit Hilfe von `gzip`.

Die nächste Option `include /etc/logrotate.d` schauen wir uns etwas weiter unten an.

Der Parameter `notifempty` lässt `logrotate` leere Log-Dateien ignorieren, während `nomail` dazu führt, dass keine Benachrichtigung per E-Mail versendet wird, und zu guter Letzt bewirkt `noolddir`, dass die Archivdateien nicht in ein separates Archivverzeichnis verschoben werden.

Danach folgt dann noch die spezifische Konfiguration für die Log-Datei `/var/log/wtmp`, mit der wir uns hier aber nicht beschäftigen wollen. Wie wir `logrotate` spezifische Einstellungen für jede Log-Datei mitgeben können, entnimmt man bei Bedarf besser der Dokumentation des Paketes.

Wichtig ist die Option `include /etc/logrotate.d`, die alle Konfigurationsdateien in `/etc/logrotate.d` einbezieht. Damit besteht die Möglichkeit, dass Pakete, die eigene Log-Dateien produzieren, direkt die Konfiguration für das Archivieren dieser Dateien mitliefern.

Im Falle unseres Webservers sollten in dem Verzeichnis `/etc/logrotate.d` die Dateien `apache2`, `mysql` und `syslog-ng` liegen. Diese stammen aus den entsprechenden Paketen:

```
gentoo ~ # qfile ‘find /etc/logrotate.d/ -type f‘
app-admin/logrotate (/etc/logrotate.d/.keep_app-admin_logrotate-0)
app-admin/syslog-ng (/etc/logrotate.d/syslog-ng)
dev-db/mysql (/etc/logrotate.d/mysql)
net-www/apache (/etc/logrotate.d/apache2)
```

‘`find /etc/logrotate.d/ -type f`‘ liefert die vollen Pfadangaben für die Dateien in `/etc/logrotate.d`, die wir als Eingabe für `qfile` brauchen, um den Ursprung der Dateien zu erfahren.

So liefert das MySQL-Paket z. B. die Archivierungsoptionen für die drei Dateien in `/var/log/mysql`, die MySQL anlegt.

```
gentoo ~ # cat /etc/logrotate.d/mysql
# Copyright 1999-2006 Gentoo Foundation
# Distributed under the terms of the GNU General Public License v2

/var/log/mysql/mysql.err /var/log/mysql/mysql.log /var/log/mysql/mysqld.e
rr {
monthly
create 660 mysql mysql
notifempty
size 5M
sharedscripts
missingok
postrotate
/bin/kill -HUP ‘cat /var/run/mysqld/mysqld.pid‘
endscript
}
```

Bleibt noch eine Datei des `app-admin/logrotate`-Paketes zu erwähnen: `/etc/cron.daily/logrotate.cron`. Dieses Skript bewirkt in Kombination mit dem `cron`-System, dass `logrotate` einmal täglich aufgerufen wird und die Log-Dateien bei Bedarf archiviert werden (siehe 16.6.4).

16.2.2 app-admin/localepurge

Die Meldungen vieler Programme sind mittlerweile in verschiedene Sprachen übersetzt, was dem Benutzer mit Hilfe der Locale-Einstellung (siehe Kapitel 8) erlaubt, die gewünschten Sprache auszuwählen.

```
gentoo ~ # export LANG="de_DE"
gentoo ~ # ls /fehlt
ls: fehlt: Datei oder Verzeichnis nicht gefunden
```

Was auf der einen Seite klare Vorteile hat, kann aber auch störend wirken: Natürlich ist es angenehm, wenn `ls` in der Muttersprache über Probleme informiert, aber warum wird z. B. gleich die finnische Übersetzung mit installiert, obwohl der Benutzer diese Sprache überhaupt nicht spricht?

Derzeit kann man, wie auf Seite 187 beschrieben, die installierten Lokalisierungen über `/etc/locale.gen` wählen. Das bezieht sich aber nur auf die Lokalisierungen der `glibc`, nicht auf die aller anderen installierten Werkzeuge. Portage ist noch nicht in der Lage, nur die gewünschten Lokalisierungen zu erhalten. Allerdings gibt es das Werkzeug `localepurge`, um unnötig verbrauchten Speicherplatz zurückzugewinnen.

```
gentoo ~ # emerge -av localepurge

These are the packages that would be merged, in order:
```

```
Calculating dependencies... done!
[ebuild  N    ] app-admin/localepurge-0.5.2  5 kB

Total: 1 package (1 new), Size of downloads: 5 kB

Would you like to merge these packages? [Yes/No] Yes
...
```

Nach der Installation des Tools muss man die Konfiguration unter `/etc/locale.nopurge` anpassen, damit sich das Programm auch ausführen lässt. Dafür müssen wir die Zeile `NEEDSCONFIGFIRST` wie im Folgenden auskommentieren:

```
########################################################
# This is the configuration file for localepurge(8).
########################################################
# Comment this to enable localepurge.
# NO COMMENT IT IF YOU ARE NOT SURE WHAT ARE YOU DOING
# THIS APP DO NOT ASK FOR CONFIRMATION

#NEEDSCONFIGFIRST

########################################################
# Uncommenting this string enables removal of localized
# man pages based on the configuration information for
# locale files defined below:

MANDELETE

########################################################
# Uncommenting this string enables display of freed disk
# space if localepurge has purged any superfluous data:

SHOWFREEDSPACE

########################################################
# Commenting out this string disables verbose output:

VERBOSE

########################################################
# You like Colors?

#NOCOLOR

########################################################
# You can use the -v -d -nc options in command linei.

########################################################
# Following locales won't be deleted from this system
# for example:
```

```
en
en_GB
de
de_DE
de_DE@euro
de_DE.UTF-8
```

Am unteren Ende der Datei trägt man dann die Lokalisierungen ein, die man *behalten* möchte, im Grunde also die Lokalisierungen, die man auch in `/etc/locale.gen` eingetragen hat.

Der nachfolgende Aufruf `localepurge` befreit uns dann von Ballast:

```
gentoo ~ # localepurge
 * localepurge: processing locale files in /usr/share/locale ...
removed '/usr/share/locale/af/LC_MESSAGES/sed.mo'
...
removed '/usr/share/locale/zh_TW/LC_MESSAGES/gettext-tools.mo'
 * localepurge: Disk space freed in /usr/share/locale: 23472K
 * localepurge: processing locale files in /usr/lib/locale ...
removed '/usr/lib/locale/en_US/LC_NAME'
...
removed '/usr/lib/locale/en_US/LC_MONETARY'
 * localepurge: Disk space freed in /usr/lib/locale: 180K
 * localepurge: processing man pages in /usr/share/man ...
 * localepurge: Disk space freed in /usr/share/man: 672K
 * localepurge: processing man pages in /usr/local/share/man ...
```

Wem der so gewonnene Speicherplatz wichtig ist, sollte den Befehl gelegentlich nach Updates ausführen. Wie wir den Prozess automatisieren, beschreiben wir im nächsten Abschnitt.

16.3 emerge erweitern: Die Datei /etc/portage/bashrc

Portage bietet zwar keine überragenden Erweiterungsmöglichkeiten, aber wenigstens lässt sich das Verhalten des Paketmanagers recht leicht beeinflussen: über die Datei `/etc/portage/bashrc`.

Die wenigsten Benutzer werden von dieser Option Gebrauch machen, denn in die Innereien des Paketmanagements eines Systems einzugreifen kann schnell fatale Folgen haben. Dennoch wollen wir den Mechanismus hier erwähnen, da er ein besonderes Feature möglich macht: die `CFLAGS` per Paket zu setzen. Das soll keine Aufforderung zum Herumexperimentieren mit `CFLAGS` sein! Es ist jedoch eine Tatsache, dass viele Gentoo-Nutzer ihre Maschine gerne soweit wie irgend möglich optimieren möchten. Da das Spiel mit den globalen `CFLAGS`, wie bereits erwähnt, bei einem laufenden

System mehr als leichtsinnig ist, wollen wir hier wenigstens die etwas sicherere Variante ansprechen.

16.3.1 Die Funktionsweise von /etc/portage/bashrc

Zunächst einmal zurück zur Arbeitsweise der Datei `/etc/portage/bashrc`. Im Normalfall existiert die Datei gar nicht, aber wenn wir sie anlegen und als ausführbar markieren, wird sie in jeder Phase der Installation aufgerufen. Am einfachsten lässt sich das an einem Beispiel verdeutlichen: Legen wir mit `nano` ein einfaches Skript in `/etc/portage/bashrc` an, das den Inhalt der Umgebungsvariablen `EBUILD_PHASE` ausgibt:

```
gentoo ~ # cat /etc/portage/bashrc
echo "*** $EBUILD_PHASE ***"
```

Jetzt installieren wie ein beliebiges, vorzugsweise kleines Paket, um zu sehen, was passiert:

```
gentoo ~ # emerge app-misc/mime-types
Calculating dependencies... done!

>>> Emerging (1 of 1) app-misc/mime-types-7 to /
*** clean ***
 * mime-types-7.tar.bz2 MD5 ;-) ...                                [ ok ]
...
 * checking mime-types-7.tar.bz2 ;-) ...                           [ ok ]
*** setup ***
*** unpack ***
>>> Unpacking source...
>>> Unpacking mime-types-7.tar.bz2 to /var/tmp/portage/mime-types-7/work
>>> Source unpacked.
*** compile ***
>>> Compiling source in /var/tmp/portage/mime-types-7/work/mime-types-7
...
>>> Source compiled.
*** test ***
>>> Test phase [not enabled]: app-misc/mime-types-7
*** install ***

>>> Install mime-types-7 into /var/tmp/portage/mime-types-7/image/ categ
ory app-misc
>>> Completed installing mime-types-7 into /var/tmp/portage/mime-types-7
/image/

***  ***
>>> Merging app-misc/mime-types-7 to /
*** preinst ***
*** preinst ***
--- /etc/
```

```
>>> /etc/mime.types
>>> Safely unmerging already-installed instance...
*** prerm ***
--- cfgpro obj /etc/mime.types
--- !empty dir /etc
*** postrm ***
*** cleanrm ***
>>> Regenerating /etc/ld.so.cache...
>>> Original instance of package unmerged safely.
*** postinst ***
>>> Regenerating /etc/ld.so.cache...
>>> app-misc/mime-types-7 merged.
*** clean ***

>>> No packages selected for removal by clean.
>>> Auto-cleaning packages...
>>> No outdated packages were found on your system.
```

Man sieht, wie die Ausgabe regelmäßig von `*** xyz ***` unterbrochen wird. An all diesen Stellen ruft Portage das `bashrc`-Skript auf und übergibt die aktuelle Phase der Installation in der Variablen `EBUILD_PHASE`. Folgende Phasen gibt es:

- `clean`
- `setup`
- `unpack`
- `compile`
- `test`
- `install`
- `preinst`
- `prerm`
- `postrm`
- `cleanrm`
- `postinst`

Die zentralen Phasen `setup`, `unpack`, `compile`, `test` und `install` haben wir bereits beim Schreiben von Ebuilds kennen gelernt. Einen genaueren Überblick bietet die Dokumentation über `man ebuild`.

16.3.2 localepurge automatisieren

Wie unter 16.2.2 beschrieben, kann man unnötige Lokalisierungen in regelmäßigen Abständen entfernen, indem man `localepurge` manuell aufruft. Über den `/etc/portage/bashrc`-Mechanismus haben wir aber nun das nötige Rüstzeug, um den Prozess auch automatisiert anzustoßen.

Der Gewinn der Aktion ist zugegebenermaßen nicht sonderlich groß, und es gibt keinen zwingenden Grund, die folgende Konfiguration zu übernehmen, aber es ist ein einfaches Beispiel, wie sich `/etc/portage/bashrc` nutzbringend einsetzen lässt.

Das Skript ist wieder denkbar einfach:

```
gentoo ~ # cat /etc/portage/bashrc
if [[ ${EBUILD_PHASE} == "postinst" ]]; then
        einfo "Running localepurge..."
        PATH="/bin:/usr/bin" localepurge
fi
```

Hier wird nur in der Phase `postinst` (`if [[ ${EBUILD_PHASE} == "postinst" ]]; ...`), also nach der Installation eines neuen Paketes das Skript `localepurge` aufgerufen, also nach beendeter Installation erst einmal aufgeräumt. Die Ausgabe von z. B. `emerge -av sys-apps/coreutils` sieht dann folgendermaßen aus:

```
gentoo ~ # emerge -av sys-apps/coreutils

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild   R   ] sys-apps/coreutils-6.4  USE="acl nls (-selinux) -static"
 0 kB

Total: 1 package (1 reinstall), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
 * Running localepurge...
 * localepurge: processing locale files in /usr/share/locale ...
/usr/share/locale/af/LC_MESSAGES/coreutils.mo entfernt
...
/usr/share/locale/zh_TW/LC_MESSAGES/coreutils.mo entfernt
 * localepurge: Disk space freed in /usr/share/locale: 4168K
 * localepurge: processing locale files in /usr/lib/locale ...
 * localepurge: processing man pages in /usr/share/man ...
 * localepurge: processing man pages in /usr/local/share/man ...
...
```

16.3.3 Paketspezifische Einstellungen

Kommen wir abschließend zu einer /etc/portage/bashrc-Variante, über die wir Einstellungen paketspezifisch vornehmen. Das dafür notwendige bashrc-Skript sieht wie folgt aus:

```
gentoo ~ # cat /etc/portage/bashrc
if [ -n "${CATEGORY}" ] && [ -n "${PN}" ]; then
    PKG_ENV_FILE="/etc/portage/package.env/${CATEGORY}/${PN}"
    if [ -r ${PKG_ENV_FILE}-${PV} ]; then
        source ${PKG_ENV_FILE}-${PV}
    elif [ -r ${PKG_ENV_FILE} ]; then
        source ${PKG_ENV_FILE}
    fi
fi
```

Die Variable EBUILD_PHASE ist nicht der einzige Wert, der dem bashrc-Skript beim Aufruf übermittelt wird. Auch die Paket-Kategorie (CATEGORY) und der Paketname (PN) sowie die zu installierende Version des Paketes (PV) werden als Variablen übergeben.

Im obigen Skript wird nun zuerst einmal geprüft, dass Kategorie und Paketname wirklich einen Wert enthalten

```
if [ -n "${CATEGORY}" ] && [ -n "${PN}" ]; ...
```

und dann getestet, ob die Datei mit angehängter Versionsnummer

```
/etc/portage/package.env/${CATEGORY}/${PN}-${PV}
```

oder auch ohne Versionsnummer

```
/etc/portage/package.env/${CATEGORY}/${PN}
```

existiert. Ist dies der Fall, wird diese Datei über source eingelesen.

Verdeutlichen wir den Vorgang an einem Beispiel: Schreibt man das oben gegebene Skript in /etc/portage/bashrc, so passiert beim emerge-Aufruf erst einmal nichts Besonderes. Da es noch kein Verzeichnis /etc/portage/package.env gibt, findet unser Skript darin auch keine lesbaren Dateien und liest sie folglich auch nicht ein.

Machen wir uns also einmal die Mühe, dort Dateien anzulegen, und aktualisieren anschließend das Mini-Paket app-misc/mime-types:

```
gentoo ~ # mkdir -p /etc/portage/package.env/app-misc
gentoo ~ # echo "[ \$EBUILD_PHASE == "unpack" ] &&  einfo \
> \"Per package call\"" > /etc/portage/package.env/app-misc/mime-types
```

Da wir die Datei /etc/portage/package.env/app-misc/mime-types erstellt haben, wird der Inhalt der Datei ausgewertet, was in der unpack-Phase zu einer kurzen Meldung (Per package call) führt:

```
gentoo ~ # emerge app-misc/mime-types
...
 * checking mime-types-7.tar.bz2 ;-) ...            [ ok ]
 * Per package call
>>> Unpacking source...
...
```

Auf diese Weise lassen sich also paketspezifisch und sogar versionsabhängige Einstellungen vornehmen. Wer z. B. mit der Performance seines Apache unzufrieden ist und durch die vage Hoffnung angetrieben wird, eine Optimierung auf Ebene des C-Compilers könne noch etwas mehr Geschwindigkeit hergeben, der kann nach diesem Verfahren die Datei /etc/portage/package.env/net-www/apache anlegen und dort z. B. Folgendes eintragen:

```
gentoo ~ # cat /etc/portage/package.env/net-www/apache
export CFLAGS="${CFLAGS} -O3"
```

Daraufhin bemüht sich der gcc-Compiler um eine maximale Optimierung in puncto Geschwindigkeit des produzierten Codes. Ob das sinnvoll ist, sei dahingestellt. Vermutlich beschleunigen andere Maßnahmen einen Apache-Server effektiver, aber dieser Thematik widmen sich andere Bücher.

16.4 Hardwaremanagement mit udev

Da Gentoo grundsätzlich die Verwendung von udev empfiehlt, wollen wir uns mit diesem System der Geräteverwaltung auch kurz auseinander setzen.

udev dient auch bei wechselnden Hardware-Profilen einer Maschine einer stabilen Verwaltung der an einen Rechner angeschlossenen Geräte. Um einen Rechner stabil konfigurieren zu können, muss man davon ausgehen, dass dieselben Geräte über eine stabile Adresse zu erreichen sind, unabhängig davon, *wie* die Geräte an den Rechner angeschlossen sind. Die Reihenfolge der Verkabelung über diverse USB-Ports sollte z. B. keinen Einfluss auf die Addressierung der Geräte haben.

Unter Linux steht standardmäßig der /dev-Dateibaum für die Geräteverwaltung zur Verfügung. Wir wollen uns hier nur kurz ansehen, wie wir die Liste der Geräte mit Hilfe von udev verwalten.

16.4.1 udev-Start

Das udev-System wird unter Gentoo im normalen Init-Systems gestartet, wenn der Nutzer tatsächlich udev für die Geräteverwaltung wählt. Dies geschieht über die Variable RC_DEVICES in der Datei /etc/conf.d/rc.

```
# Use this variable to control the /dev management behavior.
#  auto   - let the scripts figure out what's best at boot
#  devfs  - use devfs (requires sys-fs/devfsd)
#  udev   - use udev (requires sys-fs/udev)
#  static - let the user manage /dev

RC_DEVICES="udev"
```

Diese Variable kann folgende Werte annehmen:

auto
: Das Init-System versucht eigenständig herauszufinden, welche Geräteverwaltung der Benutzer wünscht. Dafür zieht es Informationen über die Kernel-Version und die installierten Pakete zu Rate.

devfs
: Immer das devfs-System wählen; sys-fs/devfsd muss installiert sein.

udev
: Immer das udev-System wählen; sys-fs/udev muss installiert sein.

static
: Es wird kein Verwaltungssystem initialisiert und der Benutzer ist selbst für die statische Verwaltung der Gerätedateien in /dev zuständig.

Wir gehen davon aus, dass der Parameter entsprechend der Empfehlung auf udev gesetzt ist. Ist dies der Fall, die Kernel-Version liegt über 2.6.0 und das sys-fs/udev-Paket ist installiert, so wird das udev-System über das Skript /lib/rcscripts/addons/udev-start.sh gestartet.

Ebenfalls über die Datei /etc/conf.d/rc können wir mit der Option RC_USE_FSTAB festlegen, ob das udev-Dateisystem mit einigen Standardoptionen auf /dev gemountet wird oder ob die Informationen der Datei /etc/fstab genutzt werden.

```
RC_USE_FSTAB="no"
```

Wer das Standardverhalten umgehen und das Dateisystem an anderer Stelle bzw. mit anderen Optionen mounten möchte, setzt RC_USE_FSTAB auf yes und trägt die eigenen Optionen in /etc/fstab ein; ist RC_USE_FSTAB

gesetzt, so lassen sich auch die Standardparameter für /proc, /sys, und /dev/pts über /etc/fstab modifizieren. Generell dürften solche Modifikationen aber nur für sehr spezielle Systeme eine Rolle spielen (siehe auch Kapitel 9.2.2 ab Seite 198).

Über udev lassen sich mittlerweile die meisten Hardware-Systeme problemlos ansprechen. Wer jedoch über besondere Hardware verfügt und einzelne Geräteeinträge in /dev vermisst, kann diese auch statisch über das Standardwerkzeug mknod hinzufügen.

Betreiben wir das /dev-Dateisystem jedoch nur über udev, entsteht hier ein Problem: Beim Neustart sind Einträge, die wir manuell über mknod hinzugefügt haben, wieder vergessen. Um aber auch diese Gerätedateien permanent zur Verfügung zu stellen, bietet Gentoo die Möglichkeit, die Einträge des dev-Verzeichnisses in einem Tar-Paket über den Neustart hinweg zu sichern.

Dazu müssen wir in der Datei /etc/conf.d/rc die Option RC_DEVICE_TARBALL auf yes (die empfohlene Standardeinstellung) setzen.

```
RC_DEVICE_TARBALL="yes"
```

16.5 Hardware-Informationen: sys-apps/x86info

Die Datei /proc/cpuinfo liefert zwar, wie schon unter 1.6.1 beschrieben, die wichtigsten Daten über den Prozessor einer Maschine, aber statt die Informationen aus dieser nicht eben übersichtlichen Datei zu klauben, bietet sich auf x86-Maschinen das Werkzeug x86info an. Die Ausgabe dieses Programms ist ein wenig besser lesbar:

```
gentoo ~ # emerge -av sys-apps/x86info

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] sys-apps/x86info-1.13  58 kB

Total: 1 package (1 new), Size of downloads: 58 kB

Would you like to merge these packages? [Yes/No] Yes
...
 * Your kernel must be built with the following options
 * set to Y or M
 *      Processor type and features ->
 *          [*] /dev/cpu/*/msr - Model-specific register support
 *          [*] /dev/cpu/*/cpuid - CPU information support
...
```

Der Warnung gegen Ende der Installation entsprechend, müssen im Kernel die beiden folgenden Optionen aktiviert sein, damit das Tool problemlos arbeiten kann:

Processor type and features
/dev/cpu/*/msr - Model-specific register support
/dev/cpu/*/cpuid - CPU information support

Unter diesen Voraussetzungen liefert das Werkzeug beim Aufruf einen knappen Überblick über die CPU-Eigenschaften der Maschine:

```
gentoo ~ # x86info
x86info v1.20.  Dave Jones 2001-2006
Feedback to <davej@redhat.com>.

Found 1 CPU
--------------------------------------------------------------------------
Family: 6 Model: 8 Stepping: 0
CPU Model : Athlon XP (Thoroughbred)[A0]
Feature flags:
 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
mmx fxsr sse
Extended feature flags:
 syscall mmxext 3dnowext 3dnow
```

Die wichtigsten Informationen sind die Familie (`Family`) und das Modell (`Athlon XP (Thoroughbred)[A0]`), mit denen sich aus den Angaben unter `http://gentoo-wiki.com/Safe_Cflags` die korrekten Einstellungen für die `CFLAGS` im Portage-System ermitteln lassen.

Die Prozessor-Flags verraten dem Experten ebenfalls, welche Optimierungen des `gcc`-Compilers sich auf der Maschine nutzen lassen. Wer die Flags etwas genauer erklärt haben möchte, fügt dem `x86info`-Aufruf die Option `--verbose` (bzw. `-v`) hinzu.

```
gentoo ~ # x86info -v
x86info v1.20.  Dave Jones 2001-2006
Feedback to <davej@redhat.com>.

Found 1 CPU
--------------------------------------------------------------------------
Family: 6 Model: 8 Stepping: 0
CPU Model : Athlon XP (Thoroughbred)[A0]
Feature flags:
        Onboard FPU
        Virtual Mode Extensions
        Debugging Extensions
        Page Size Extensions
        Time Stamp Counter
```

```
        Model-Specific Registers
        Physical Address Extensions
        Machine Check Architecture
        CMPXCHG8 instruction
        Onboard APIC
        SYSENTER/SYSEXIT
        Memory Type Range Registers
        Page Global Enable
        Machine Check Architecture
        CMOV instruction
        Page Attribute Table
        36-bit PSEs
        MMX support
        FXSAVE and FXRESTORE instructions
        SSE support

Extended feature flags:
 syscall mmxext 3dnowext 3dnow
```

`x86info` bietet noch einige exotischere Features, um z. B. die Register der CPU anzuzeigen, aber auf diese wollen wir hier nicht weiter eingehen. Sie sind für die Bestimmung der `CFLAGS` irrelevant.

16.6 cron

Es gibt eine Reihe von Prozessen, die in regelmäßigen Abständen auf dem Rechner laufen sollten, z. B. Sicherheitschecks, das Aufräumen temporärer Dateien oder Updates. Insbesondere bei Server-Systemen werden solche Prozesse im Regelfall automatisch angestoßen und bedürfen keiner Nutzerinteraktion. Voraussetzung dafür ist ein `cron`-System, das diese Aktionen zeitabhängig startet.

Wie bereits im Rahmen der Installation (Seite 57) erwähnt, bietet Gentoo verschiedene Cron-Systeme an; derzeit sind gleich fünf im Angebot: `sys-process/anacron`, `sys-process/bcron`, `sys-process/dcron`, `sys-process/fcron` und `sys-process/vixie-cron`.

`sys-process/anacron` nutzt allerdings nicht das Standardsystem, und `sys-process/bcron` ist noch recht neu und nicht als stabil markiert. Beide Pakete sollte man nur einsetzen, wenn man auf deren besondere Features nicht verzichten kann.

Die Pakete `sys-process/dcron`, `sys-process/fcron` und `sys-process/vixie-cron` haben eins gemeinsam: Sie basieren auf `sys-process/cronbase` und liefern dem Benutzer darüber eine einheitliche Schnittstelle, die auch von anderen Distributionen bekannt sein dürfte.

16.6.1 sys-process/cronbase

Schauen wir uns an, welche Dateien zu `sys-process/cronbase` gehören, erschließt sich bereits dessen Konzept:

```
gentoo ~ # qlist sys-process/cronbase
/usr/sbin/run-crons
/usr/share/doc/cronbase-0.3.2/README.gz
/etc/cron.hourly/.keep
/etc/cron.daily/.keep
/etc/cron.weekly/.keep
/etc/cron.monthly/.keep
/var/spool/cron/.keep
/var/spool/cron/lastrun/.keep
```

Das Skript `/usr/sbin/run-crons` führt die einzelnen Skripte in den Verzeichnissen `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly` und `/etc/cron.monthly` aus. Diese Verzeichnisse finden sich auch bei anderen Distributionen.

In jedem dieser Verzeichnisse lassen sich wiederum ausführbare Skripte ablegen, die das `cron`-System entsprechend dem Verzeichnisnamen in regelmäßigen Abständen stündlich, täglich, wöchentlich oder monatlich ausführt.

Zuletzt sorgt das auf der `cronbase` aufsetztende `cron`-System dafür, dass `/usr/sbin/run-crons` auch wirklich in regelmäßigen Abständen aufgerufen wird. Wie das genau vor sich geht beschreiben wir im folgenden Abschnitt anhand des `vixie-cron`-Systems.

Auch wenn `cronbase` nur einen Bruchteil der Funktionen nutzt, die ein `cron`-System im Normalfall bietet, ist es als Grundlage sinnvoll, da es sicher 95% der Anwendungsfälle im Administratorenalltag abdeckt. Wem es also genügt, Skripte in den vier oben genannten Zeitabständen auszuführen, wählt als auf `sys-process/cronbase` aufsetztendes `cron`-System einfach den von Gentoo empfohlenen Standard `sys-process/vixie-cron`.

16.6.2 Die einfachste Lösung: sys-process/vixie-cron

Das Paket ist in Sachen Konfiguration einfach zu handhaben. Nach der Installation startet man den `cron`-Service und fügt ihn, wenn gewünscht, dem `default`-Runlevel hinzu:

```
gentoo ~ # emerge -av sys-process/vixie-cron
...
gentoo ~ # /etc/init.d/vixie-cron start
gentoo ~ # rc-update add vixie-cron default
```

Da sys-process/vixie-cron eine systemweite cron-Konfiguration in /etc/crontab akzeptiert, funktioniert damit das oben beschriebene sys-process/cronbase-System sofort. Ein Blick in die Datei /etc/crontab verrät den Ablauf:

```
gentoo ~ # cat /etc/crontab
# for vixie cron

# Global variables
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# check scripts in cron.hourly, cron.daily, cron.weekly and cron.monthly
0  *  * * *      root  rm -f /var/spool/cron/lastrun/cron.hourly
1  3  * * *      root  rm -f /var/spool/cron/lastrun/cron.daily
15 4  * * 6      root  rm -f /var/spool/cron/lastrun/cron.weekly
30 5  1 * *      root  rm -f /var/spool/cron/lastrun/cron.monthly
*/10  *  * * *   root  test -x /usr/sbin/run-crons && /usr/sbin/run-crons
```

Die letzten fünf Zeilen sind für den zeitlichen Ablauf des sys-process/cronbase-Systems verantwortlich. Die Einträge bestimmen in cron-üblicher Manier die gewünschten Zeitabstände. Die erste Spalte definiert die Minute, die zweite die Stunden, die dritte die Tage des Monats und die vierte die Monate. Die fünfte Spalte nennt Wochentage für wöchentliche Aufgaben.

Die erste Zeile definiert über `0 * * * *` eine stündliche Aktion: in der ersten Spalte die Minute `0` und in allen anderen Spalten jeder Zeitpunkt über `*`. So führt das cron-System jeweils zur vollen Stunde die in der siebten Spalte genannte Aktion (`rm -f /var/spool/cron/lastrun/cron.hourly`) aus. Die sechste Spalte bezeichnet den Benutzer, unter dessen Kennung das Skript ausgeführt wird. Die zweite Zeile spezifiziert für das „Tages-Skript“ mit `1 3 * * *` die Uhrzeit `3:01` an jedem Tag der Woche.

Der Ablauf sieht so aus, dass der cron-Prozess zu den angegebenen Zeiten die Information über einen ausgeführten cron-Job löscht, indem er die entsprechende Indikator-Datei /var/spool/cron/lastrun/cron.{hourly,daily,weekly,monthly} entfernt. Alle zehn Minuten (siehe `*/10 * * * *` in Zeile fünf) wird dann das Skript /usr/sbin/run-crons aus dem cronbase-Paket ausgeführt. Dieses überprüft, ob eine der /var/spool/cron/lastrun/cron.*-Dateien fehlt; sollte dies der Fall sein, führt es die Skripte in dem entsprechenden /etc/cron.*-Verzeichnis aus.

Wem die /etc/cron.*-Verzeichnisse für das Management der notwendigen Aufgaben genügen, muss sich an dieser Stelle nicht weiter mit dem cron-System auseinander setzen, sondern fügt bei Bedarf einfach ein Skript in die /etc/cron.*-Verzeichnisse ein.

16.6.3 sys-process/dcron und sys-process/fcron

Sowohl `sys-process/dcron` als auch `sys-process/fcron` unterstützen keine zentrale, systemweite `/etc/crontab`-Datei und benötigen damit für die Konfiguration zumindest einen weiteren Schritt.

Beide Pakete liefern die notwendigen Informationen zum Abschluss der Installation:

```
gentoo ~ # emerge -av sys-process/dcron

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] sys-process/dcron-3.2  22 kB

Total: 1 package (1 new), Size of downloads: 22 kB

Would you like to merge these packages? [Yes/No] Yes
...
 * To activate /etc/cron.{hourly|daily|weekly|monthly} please run:
 *  crontab /etc/crontab
 *
 * !!! That will replace root's current crontab !!!
 *
 * You may wish to read the Gentoo Linux Cron Guide, which can be
 * found online at:
 *     http://www.gentoo.org/doc/en/cron-guide.xml
```

Da beide nur Benutzerspezifische `crontab`-Informationen akzeptieren und `/etc/crontab` nicht automatisch ausgeführt wird, ist `/etc/crontab` einmal für einen spezifischen Nutzer zu installiert.

```
gentoo ~ # crontab /etc/crontab
```

Dieses Kommando, als `root` ausgeführt, installiert die Anweisungen aus `/etc/crontab` für den Benutzer `root`. Die Angaben in der Standard-`crontab` ähneln stark der Syntax von `sys-process/vixie-cron`:

```
gentoo ~ # cat /etc/crontab
# /etc/crontab

# fcron || dcron:
# This is NOT the system crontab! fcron and dcron do not support a syste
m crontab.
# to get /etc/cron.{hourly|daily|weekly|montly} working with fcron or
dcron do
# crontab /etc/crontab
# as root.
```

```
# NOTE: This will REPLACE root's current crontab!!


# check scripts in cron.hourly, cron.daily, cron.weekly and cron.monthly
0  *  * * *      rm -f /var/spool/cron/lastrun/cron.hourly
0  3  * * *      rm -f /var/spool/cron/lastrun/cron.daily
15 4  * * 6      rm -f /var/spool/cron/lastrun/cron.weekly
30 5  1 * *      rm -f /var/spool/cron/lastrun/cron.monthly
*/15 * * * *     test -x /usr/sbin/run-crons && /usr/sbin/run-crons
```

Hier fehlt im Vergleich zu `sys-process/vixie-cron` nur die Angabe zum Benutzer, da diese ja durch die benutzerspezifische Installation der `sys-process/cronbase`-Informationen vorgegeben ist.

Es gibt übrigens Pakete, die `sys-process/cronbase` zwingend erfordern, da sie bei der Installation Skripte in den `/etc/cron.*`-Verzeichnissen ablegen. Wer also `sys-process/dcron` oder `sys-process/fcron` installiert, darf nicht vergessen, die `crontab` auch tatsächlich zu aktivieren. Die einmal erzeugten Einträge in der `root-crontab` sollten wir darum auch nicht einfach als „überflüssig“ entfernen.

Wer das `cron`-System über die Funktionalität der `sys-process/cronbase` hinaus nutzen möchte, sei auf die Gentoo-Referenz[1] verwiesen. Diese gibt einen detaillierten Überblick über die Unterschiede in Bedienung und Konfiguration zwischen den Paketen.

16.6.4 Beispielszenarien

Abschließend wollen wir auf Basis des `sys-process/cronbase`-Systems einige Beispielszenarien für das `cron`-System auf einer Gentoo-Maschine vorstellen, wobei wir uns der Einfachheit halber auf das Verzeichnis `/etc/cron.daily`, also tägliche Aufgaben beschränken. Natürlich können Sie die Zeitintervalle beliebig anpassen und die Skripte in anderen Verzeichnissen platzieren.

Die meisten der im Folgenden besprochenen Skripte müssen Sie manuell mit `nano` in den entsprechenden Ordnern erstellen; andere – wie `slocate` oder `logrotate.cron` – installiert `emerge` als Teil entsprechender Pakete (`sys-apps/slocate` bzw. `app-admin/logrotate`).

Aufräumen mit eclean

Es empfiehlt sich, das Verzeichnis mit den heruntergeladenen Quellarchiven in regelmäßigen Abständen mit Hilfe von `eclean` (siehe Kapitel 11.1.4) von unnötigem Ballast zu befreien, denn einige dieser Quellarchive haben einen beträchtlichen Umfang und verschwenden wertvollen Speicherplatz.

[1] `http://www.gentoo.org/doc/en/cron-guide.xml`

Folgendes Skript in /etc/cron.daily/eclean löscht überflüssige Dateien:

```
gentoo ~ # cat /etc/cron.daily/eclean
#! /bin/sh

echo
echo "*****************************************************"
echo "* Running eclean"
echo "*****************************************************"
echo

eclean --nocolor distfiles
```

Die echo-Zeilen sind nicht zwingend notwendig, erhöhen aber die Übersicht, wenn wir uns den Bericht über cron-Aktivitäten per Mail zuschicken lassen.

Tägliche Synchronisation

Wir haben schon in Kapitel 10.6 den Update-Zyklus angesprochen und wollen hier eine Möglichkeiten beschreiben, die Aktualisierung zu automatisieren.

Es kann auf keinen Fall schaden, den Portage-Baum täglich zu synchronisieren, um stets über Updates informiert zu sein. Man könnte also direkt emerge --sync über ein Skript in /etc/cron.daily aufrufen:

```
gentoo ~ # cat /etc/cron.daily/esync
#! /bin/sh

echo
echo "*****************************************************"
echo "* Running esync"
echo "*****************************************************"
echo

emerge --sync --quiet
```

Wer sich das Ergebnis der täglichen Aktualisierung zumailen lässt sollte mit der Option --quiet die Ausgabe der rsync-Meldungen unterdrücken.

Beim Einsatz von esearch (siehe Kapitel 13.2) sollte sollte man auch automatisiert die esearch-Datenbank aktualisieren und folgende Zeile anhängen:

```
eupdatedb --quiet --nocolor
```

Wir verwenden auch hier wieder die Option `--quiet` und dazu `--nocolor`, da die Farbcodes in der Ausgabe des Cron-Logs problematisch sein können.

Wer `eix` (siehe Kapitel 13.5) als Suchwerkzeug nutzt muss nach der Synchronisation des Portage-Baums ebenfalls die Datenbank dieses Werkzeugs mit `update-eix` aktualisieren und hängt folgende Zeile an:

```
update-sync --quiet
```

Wer ausschließlich `esearch` bzw. `eix` verwendet kann sich auch für das entsprechende der kombinierten Skripte `esync` bzw. `eix-sync` entscheiden. Jedes kombiniert den Aufruf `emerge --sync` mit der Aktualisierung der jeweiligen Datenbank.

Um das System darüber hinaus täglich mit den neuesten Paketversionen zu bestücken, muss der Aktualisierung erneut `emerge` folgen, so dass wir das Kommando zur Systemaktualisierung von Seite 218 an unser Skript anfügen:

```
emerge -uND world
```

Wie schon in Kapitel 10.6 erwähnt, finden sich gelegentlich auch Fehler in den Paketen, die eine Installation verhindern. Beim manuellen Aufruf von `emerge -uND world` wird `emerge` den Installations-Vorgang bei problematischen Paketen abbrechen, so dass man den Fehler beheben und `emerge -uND world` erneut aufrufen kann. Bei der automatischen Installation führen solche Fehler dazu, dass `emerge` alle nachfolgenden Pakete nicht mehr bearbeiten kann.

Hier helfen zwei Optionen von `emerge`, die wir bislang noch nicht genutzt haben: `--resume` und `--skipfirst`. `--resume` greift den `emerge`-Prozess dort wieder auf, wo er zuletzt unterbrochen wurde, und `--skipfirst` sorgt – stets in Kombination mit `--resume` – dafür, dass `emerge` das erste Paket auf der verbleibenden Aktualisierungsliste überspringt.

Daraus ergibt sich folgendes Bash-Skript:

```
emerge -uDN world || until emerge --resume --skipfirst ; do emerge
--resume --skipfirst ; done
```

Die Zeile bewirkt, dass zunächst einmal `emerge -uDN world` durchlaufen wird. Bei Erfolg passiert weiter gar nichts. Sollte es zu einem Fehler kommen, wird der Teil hinter der ODER-Anweisung (`||`) ausgeführt, und zwar so oft (`do ...; done`), bis (`until`) `emerge --resume --skipfirst` keinen Fehler mehr zurück gibt. Auf diese Weise werden alle problematischen Pakete übersprungen.

Zugegebenermaßen ist dies ein recht rabiates Vorgehen, zumal es völlig außer Acht lässt, dass manche Pakete auch eine Aktualisierung der Konfiguration erfordern (siehe Kapitel 10.3). Auch der Tatsache, dass eine Aktualisierung Abhängigkeiten zwischen Bibliotheken stören kann und wir vor einer Fortsetzung der Installation `revdep-rebuild` laufen lassen sollten (siehe Kapitel 10.4.4), wird das Verfahren nicht gerecht.

Es gibt mittlerweile ausgefeiltere Skripte zur Aktualisierung, und es bleibt zu hoffen, dass diese in Portage selbst einfließen. Bis dahin empfiehlt das manuelle Update in größeren Zeitabständen.

layman synchronisieren

Ähnlich lassen sich auch über `layman` verwaltete Overlays automatisch aktualisieren. Der Befehl `layman -S` synchronisiert alle installierten Overlays:

```
gentoo ~ # cat /etc/cron.daily/layman_update
#! /bin/sh

echo
echo "*****************************************************"
echo "* Running layman -S"
echo "*****************************************************"
echo

layman -S
```

Die Ausgabe dieses Befehls kann bei vielen Overlays allerdings unübersichtlich werden. Wer mag kann die Ausgabe mit `layman -S > /dev/null` an `/dev/null` schicken und damit unterdrücken.

Sicherheitscheck mit glsa-check

Den Sicherheitsstatus einer Maschine zu überprüfen ist eine Aufgabe, die das Cron-System ebenfalls täglich abhandeln sollte. Dazu lässt sich das in Kapitel 11.1.2 besprochene Skript `glsa-check` verwenden. Ein mögliches Cron-Skript könnte z. B. so aussehen:

```
gentoo ~ # cat /etc/cron.daily/glsa-check
#! /bin/sh

echo
echo "*****************************************************"
echo "* Running glsa-check"
echo "*****************************************************"
echo

glsa-check --list --nocolor | grep -v "\[U\]"
```

Der grep-Befehl filtert nur die wirklich bestehenden Sicherheitslücken aus der glsa-check-Ausgabe, denn normalerweise listet das Werkzeug alle bekannten GLSA auf. Das würde aber die zusammengefasste Ausgabe der täglichen cron-Aufgaben sehr unübersichtlich gestalten.

locate-Datenbank aktualisieren

sys-apps/slocate ist eines der Pakete, das bei der Installation ein Skript in /etc/cron.daily legt. Um über slocate eine Datei möglichst schnell im Dateisystem zu finden, hält das Paket eine interne Datenbank vor, die wir regelmäßig aktualisieren sollten. Genau diesem Zweck dient das Skript /etc/cron.daily/slocate, das den Befehl updatedb aufruft:

```
gentoo ~ # cat /etc/cron.daily/slocate
#! /bin/sh

if [ -x /usr/bin/updatedb ]
then
        if [ -f /etc/updatedb.conf ]
        then
                nice /usr/bin/updatedb
        else
                nice /usr/bin/updatedb -f proc
        fi
fi
```

Auch diesem Standard-Skript kann man echo-Zeilen wie in den übrigen Beispielen hinzufügen. Im Normalfall liefert updatedb zwar keine Ausgabe, aber sollten Fehler auftreten, lassen sich diese besser zuordnen, wenn sie in der Ausgabe aller /etc/cron.daily-Skripte mit einer eindeutigen Kopfzeile versehen sind.

Log-Dateien rotieren

In der gleichen Weise installiert das Paket app-admin/logrotate ein Skript namens logrotate.cron in /etc/cron.daily. Es dient dazu, die Log-Dateien eines Systems in regelmäßigen Abständen zu archivieren und so zu verhindern, dass diese Dateien ins Unendliche wachsen.

Um diesen Prozess in definierten Zeitabständen ablaufen zu lassen, verlässt sich logrotate vollständig auf ein installiertes cron-System. Der entsprechende Befehl im Skript ist denkbar einfach:

```
gentoo ~ # cat /etc/cron.daily/logrotate.cron
#! /bin/sh

/usr/sbin/logrotate /etc/logrotate.conf
```

16.7 Schnelleres Kompilieren

Einer der gravierenden Nachteile von Gentoo ist und bleibt das Kompilieren der Pakete. Verständlicherweise versuchen sowohl Entwickler als auch Benutzer die für diesen Vorgang benötigte Zeit soweit wie möglich zu reduzieren.

Der Spielraum ist allerdings sehr begrenzt, aber wir wollen hier doch einige Möglichkeiten für einen kleinen Zeitgewinn beschreiben.

16.7.1 ccache

Als es um die Datei `/etc/make.conf` in Kapitel 6.3.3 ging, war auf Seite 152 bereits von dem Feature `ccache` die Rede.

Es aktiviert einen sogenannten *Compiler-Cache*. Mit dessen Hilfe behält der `gcc`-Compiler eine Kopie jeder übersetzten C-Datei und kann beim erneuten Kompilieren derselben Datei unter identischen Bedingungen (CPU, `CFLAGS` etc.) direkt auf das Ergebnis zugreifen.

Wie sinnvoll das Verfahren ist und welche Zeitersparnis daraus resultiert, wollen wir erst am Ende des Abschnitts diskutieren und zunächst einmal den Cache selber in Betrieb nehmen. Das ist recht einfach und darum auch zu empfehlen.

ccache einrichten

Um den Compiler-Cache in Betrieb zu nehmen, ist das Paket `dev-util/ccache` zu installieren, das das Programm `ccache` bereitstellt:

```
gentoo ~ # emerge -av dev-util/ccache

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N    ] dev-util/ccache-2.4-r6  0 kB

Total: 1 package (1 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No] Yes
...
```

Damit ist der Cache auch schon fast einsatzbereit. Wir müssen zuvor in der Datei `/etc/make.conf` der Variablen `FEATURES` noch den Wert `ccache` hinzufügen (vgl. Seite 152).

```
FEATURES="sandbox parallel-fetch strict distlocks ccache"
```

Dazu sollten wir ebenfalls in `/etc/make.conf` über `CCACHE_SIZE` die Größe des Caches festlegen:

```
CCACHE_SIZE="2G"
```

Standardmäßig ist er auf 512 MB festgelegt, empfohlen werden 2 GB (die wir, wie oben angegeben, mit `2G` bestimmen).

Den Ort des Caches legt `CCACHE_DIR` per Default auf `/var/tmp/ccache`, und wir wollen diesen Wert auch nicht verändern.

ccache testen

Testen wir den Effekt des Compiler-Cache am schon bekannten `dev-libs/openssl`-Beispiel! Für dieses Paket hatten wir uns auf Seite 261 mit `qlop` die Compile-Zeiten angesehen:

```
gentoo ~ # qlop -H -g dev-libs/openssl
openssl: Tue Jan 29 20:16:51 2008: 9 minutes, 8 seconds
openssl: Wed Jan 30 08:18:00 2008: 15 minutes, 21 seconds
openssl: Wed Jan 30 10:09:51 2008: 8 minutes, 40 seconds
openssl: Wed Jan 30 14:23:20 2008: 12 minutes, 53 seconds
openssl: 4 times
```

Schauen wir uns zunächst den Zustand des (noch) leeren Caches an. Dafür verwenden wir das oben erwähnte `ccache` und geben ihm mit `CCACHE_DIR="/var/tmp/ccache"` die Position unseres Caches an. Die Option `-s` liefert die Statistik der unter `/var/tmp/ccache` gespeicherten Compiler-Ergebnisse:

```
gentoo ~ # CCACHE_DIR="/var/tmp/ccache" ccache -s
cache directory                     /var/tmp/portage
cache hit                              0
cache miss                             0
files in cache                         0
cache size                             0 Kbytes
max cache size                       2.0 Gbytes
```

Wir kompilieren bzw. installieren `dev-libs/openssl` nun einfach noch einmal und schauen uns die Ergebnisse des Laufs an:

```
gentoo ~ # emerge dev-libs/openssl
...
gentoo ~ # qlop -H -g dev-libs/openssl
openssl: Tue Jan 29 20:16:51 2008: 9 minutes, 8 seconds
openssl: Wed Jan 30 08:18:00 2008: 15 minutes, 21 seconds
openssl: Wed Jan 30 10:09:51 2008: 8 minutes, 40 seconds
```

```
openssl: Wed Jan 30 14:23:20 2008: 12 minutes, 53 seconds
openssl: Sat Feb  2 08:11:04 2008: 13 minutes, 19 seconds
openssl: 5 times
```

Der Lauf ist offensichtlich nicht schneller geworden, allerdings war das auch nicht zu erwarten, denn schließlich haben wir den Cache gerade erst in Betrieb genommen und noch überhaupt keine `gcc`-Ergebnisse zwischengespeichert.

Vielmehr ist es also erfreulich, dass das Aktivieren des Caches keinen negativen Effekt hatte. Immerhin haben wir nun auch die ersten Daten gesammelt und schauen uns die Cache-Statistik an:

```
gentoo ~ # CCACHE_DIR="/var/tmp/ccache" ccache -s
cache directory                     /var/tmp/ccache
cache hit                              0
cache miss                           610
called for link                        1
not a C/C++ file                      54
unsupported compiler option           51
no input file                          2
files in cache                      1220
cache size                           4.8 Mbytes
max cache size                       2.0 Gbytes
```

`gcc` und `ccache` haben also bei der letzten Installation 1220 Ergebnisse zwischengespeichert. Wir prüfen, ob diese Zwischenergebnisse den nächsten Lauf beschleunigen:

```
gentoo ~ # emerge dev-libs/openssl
...
gentoo ~ # qlop -H -g dev-libs/openssl
openssl: Tue Jan 29 20:16:51 2008: 9 minutes, 8 seconds
openssl: Wed Jan 30 08:18:00 2008: 15 minutes, 21 seconds
openssl: Wed Jan 30 10:09:51 2008: 8 minutes, 40 seconds
openssl: Wed Jan 30 14:23:20 2008: 12 minutes, 53 seconds
openssl: Sat Feb  2 08:11:04 2008: 13 minutes, 19 seconds
openssl: Sat Feb  2 08:30:59 2008: 7 minutes, 51 seconds
openssl: 6 times
```

Jetzt macht sich der Cache deutlich bemerkbar: Wir haben ca. fünf Minuten eingespart und damit den Zeitbedarf für die Installation des Pakets um ein Drittel reduziert.

Der Effekt von ccache

Das klingt gut, aber man darf eines nicht vergessen: Wir haben das Paket einfach zweimal kompiliert und installiert, und das simuliert nicht gerade eine Standardsituation.

Natürlich kann es vorkommen, dass man ein Paket zweimal kurz hintereinander kompiliert, weil z. B. im ersten Lauf ein bestimmtes USE-Flag vergessen wurde; üblich sind Neuinstallationen aber nur nach einigen Wochen, nämlich wenn eine neue Version verfügbar ist.

Versionsunterschiede bedingen wiederum auch Veränderungen am Code, so dass nicht mehr alle Ergebnisse aus dem Cache verwendbar sind. Außerdem kann es sein, dass wir in der Zwischenzeit so viele Pakete neu kompiliert haben, dass der letzte Installationsprozess aus Platzgründen schon aus dem Cache entfernt wurde. Aus diesen Gründen fallen Einsparungen im tatsächlichen Betrieb deutlich geringer aus als uns das eingesparte Drittel bei `dev-libs/openssl` hat hoffen lassen.

Andererseits ist das Paket so trivial einzurichten, dass auch nichts dagegen spricht, `dev-util/ccache` einzusetzen, jedenfalls nicht, wenn noch zwei GB Platz auf der Festplatte zur Verfügung stehen.

16.7.2 distcc einrichten

Während jeder Nutzer einen Compiler-Cache einrichten kann, um die Installationszeiten zu reduzieren, fällt das beim verteilten Kompilieren schwerer. Die eigentliche Installation bzw. Konfiguration ist ähnlich simpel, aber es bedarf zwingend mehrerer Gentoo-Rechner – und das ist wohl nicht immer gegeben.

Darüber hinaus müssen die Rechner in gewissen Grenzen binär kompatibel sein, d. h.

- die Rechner sollten die gleiche Architektur bzw. den gleichen `CHOST`-Wert haben (z. B. `CHOST="i686-pc-linux-gnu"`).
- die verwendete `gcc`-Version sollte auf allen Rechnern identisch sein.

Wir sprechen bei beiden Bedingungen von „sollte“, denn tatsächlich lassen sie sich durch eine deutlich komplexere Konfiguration umgehen; allerdings können dadurch auch Fehler in den erstellten Programmen auftreten.

Wir beleuchten hier also nur den einfachsten Fall und verweisen Experimentierfreudige auf die einschlägige Gentoo-Dokumentation[2].

Ersteinmal installieren wir das Paket `sys-devel/distcc` auf allen beteiligten Rechnern:

```
gentoo ~ # emerge -av sys-devel/distcc

These are the packages that would be merged, in order:
```

[2] `http://www.gentoo.org/doc/en/distcc.xml`

```
Calculating dependencies... done!
[ebuild  N    ] sys-devel/distcc-2.18.3-r10  USE="-gnome -gtk -ipv6 (-se
linux)" 334 kB

Total: 1 package (1 new), Size of downloads: 334 kB

Would you like to merge these packages? [Yes/No] Yes
...
```

Es folgt, wie oben für den Compiler-Cache, die Aktivierung des Portage-Features `distcc` (vgl. Seite 152): Wir fügen die Option `ccache` auf allen verwendeten Rechnern zur Variable `FEATURES` hinzu:

```
FEATURES="sandbox parallel-fetch strict distlocks ccache distcc"
```

Indem wir mehrere Rechner zum Kompilieren nutzen, fügen wir unserem System quasi zusätzliche CPUs hinzu. Wir hatten schon in Abschnitt 6.3.1 die Variable `MAKEOPTS` beschrieben und festgelegt, dass sie die Option `-j` mit der Anzahl der verfügbaren CPUs plus eins enthalten soll. Entsprechend müssen wir diesen Wert ebenfalls auf allen Maschinen in unserem Compiler-Netzwerk anpassen. Nehmen wir einmal an, wir verwenden nur zwei Rechner:

```
MAKEOPTS="-j3"
```

Damit sind wir fast fertig, müssen aber noch zusehen, dass die verwendeten Rechner miteinander kommunizieren. Dazu verwenden wir zum einen das Programm `distcc-config`. Diesem teilen wir auf jedem Rechner mit der Option `--set-hosts` die IP-Adressen aller Hosts in unserem Compiler-Netzwerk mit:

```
gentoo ~ # distcc-config --set-hosts "192.168.178.2 192.168.178.3"
```

Außerdem müssen die Rechner des Compiler-Netzwerkes in der Lage sein, automatisch Compile-Jobs auszutauschen. Dafür sorgt der `distcc`-Daemon (`distccd`). Damit dieser nicht einfach von jedem Rechner Aufträge entgegennimmt, müssen wir die als sicher eingestuften IP-Adressen in der Datei `/etc/conf.d/distcc` eintragen und fügen einzelne Hosts bzw. ganze Netzwerke mit der Option `--allow` *IP* bzw. *IP/SUBNETMASK* zur Variablen `DISTCCD_OPTS` hinzu.

In Beispiel würden wir auf der Maschine mit der IP `192.168.178.2` in der Datei `/etc/conf.d/distcc` die Variable `DISTCCD_OPTS` folgendermaßen setzen:

```
DISTCCD_OPTS="${DISTCCD_OPTS} --allow 192.168.178.3"
```

Entsprechend findet sich dann auf der Maschine mit der IP `192.168.178.3` der passende Eintrag:

```
DISTCCD_OPTS="${DISTCCD_OPTS} --allow 192.168.178.2"
```

Damit können wir den Daemon starten und sollten ihn auch mit `rc-update` unserer Bootsequenz hinzufügen:

```
gentoo ~ # /etc/init.d/distccd start
gentoo ~ # rc-update add distccd default
```

Das System ist damit korrekt konfiguriert. Während man auf der einen Maschine ein Paket kompiliert, kann man nun auf der zweiten die von `distccd` angestoßene `gcc`-Aktivität beobachten:

```
gentoo ~ # ps -ax --forest
28118 ?        SNs    0:00 /usr/bin/distccd --pid-file /var/run/distccd/
29934 ?        SN     0:00  \_ /usr/bin/distccd --pid-file /var/run/dist
29950 ?        SN     0:00  \_ /usr/bin/distccd --pid-file /var/run/dist
30631 ?        SN     0:00  |   \_ i686-pc-linux-gnu-gcc -O2 -mtune=i686
30632 ?        RN     0:00  |       \_ /usr/libexec/gcc/i686-pc-linux-gn
30633 ?        SN     0:00  |       \_ /usr/lib/gcc/i686-pc-linux-gnu/3.
29984 ?        SN     0:00  \_ /usr/bin/distccd --pid-file /var/run/dist
30634 ?        SN     0:00  |   \_ i686-pc-linux-gnu-gcc -O2 -mtune=i686
30635 ?        RN     0:00  |       \_ /usr/libexec/gcc/i686-pc-linux-gn
30636 ?        SN     0:00  |       \_ /usr/lib/gcc/i686-pc-linux-gnu/3.
30010 ?        SN     0:00  \_ /usr/bin/distccd --pid-file /var/run/dist
30637 ?        RN     0:00      \_ /usr/bin/distccd --pid-file /var/run/
```

Beim Kompilieren wird sich nun ein deutlicher Zeitvorteil bemerkbar machen.

16.8 Interaktion mit dem Gentoo-Projekt

Gentoo ist freie Software, ein offenes Projekt und bietet eine Vielzahl von Schnittstellen zur Interaktion. Vor allem wenn Sie Hilfe benötigen, sollte es keine Schwierigkeit geben, diese auch zu finden.

Wir wollen hier nur die wesentlichen Anlaufstellen nennen und zeigen, wie Sie auf der Kommandozeile mit Hilfe von IRC möglichst zeitnah Unterstützung bei einem Problem bekommen.

Zu guter Letzt beschreiben wir, wie Sie Fehler in der Bug-Datenbank von Gentoo einreichen. Keine Sorge, früher oder später werden Sie einen finden.

16.8.1 Hilfe suchen und finden

Die professionelle Dokumentation zu allen möglichen Themenbereichen rund um Gentoo wird vom *Gentoo Documentation Project* erstellt und gepflegt. Als Einstiegspunkt dient hier entweder die englische Seite des Projekts[3] oder die deutsche Übersetzung[4]. Die Qualität dieser Dokumente ist sehr hoch und die meisten Artikel werden regelmäßig aktualisiert.

Findet sich hier keine Antwort auf Ihre Fragen, sollten Sie im zweiten Schritt das englische[5] bzw. das deutsche Gentoo-Wiki[6] konsultieren.

Bleiben nach wie vor Fragen offen, bieten sich interaktive Kommunikationssysteme an. Hier sind vor allem das Forum[7] und die Mailinglisten[8] zu nennen.

Bevor Sie selbst Fragen stellen, sollten Sie unbedingt versuchen, in Forums-Beiträgen bzw. den Nachrichten der Mailingliste nach passenden Schlagwörtern zu suchen, um ähnliche, vielleicht schon beantwortete Fragen zu finden; so vermeiden Sie unnötige Wiederholungen.

Manchmal benötigt man auch sofort eine Antwort auf seine Frage, und für diesen Fall bietet sich IRC (*Internet Relay Chatting*) an.

16.8.2 IRC

IRC ist ein textbasiertes Chat-System, das sich auch von der Kommandozeile bedienen lässt. So bietet z. B. schon die LiveDVD den IRC-Client `net-irc/irssi`, über den Sie bereits während der Installation Fragen im Netz stellen können. Voraussetzung ist natürlich, dass zumindest die Netzwerkkarte funktioniert.

In einem neu installierten System ist der Client zunächst zu installieren:

```
gentoo ~ # emerge -av net-irc/irssi

These are the packages that would be merged, in order:

Calculating dependencies... done!
[ebuild  N   ] dev-libs/glib-2.12.9  USE="-debug -doc -hardened" 0 kB
[ebuild  N   ] net-irc/irssi-0.8.10-r4  USE="ipv6 perl ssl -socks5" 0 kB

Total: 2 packages (2 new), Size of downloads: 0 kB

Would you like to merge these packages? [Yes/No]
```

[3] http://www.gentoo.org/doc/en/index.xml
[4] http://www.gentoo.org/doc/de/index.xml
[5] http://gentoo-wiki.com/Main_Page
[6] http://de.gentoo-wiki.com/Hauptseite
[7] http://forums.gentoo.org/
[8] http://www.gentoo.org/main/en/lists.xml

Wir starten den Client mit `irssi` (wenn irgend möglich *nicht* als Administrator unter dem `root`-Account) und landen in einer grafischen Benutzeroberfläche, in der wir uns mit einem IRC-Server verbinden. Da sich alle Gentoo-spezifischen Kanäle auf Freenode befinden, lautet das Kommando `/connect irc.freenode.net`.

Anschließend sollten Sie sich einen Namen mit `/nick` *`meinname`* geben und betreten dann einen der Gentoo-Kanäle[9] mit `/join` *`#kanal`*.

Im englische Kanal (`/join #gentoo`) tummeln sich meist mehrere hundert Nutzer, während es im deutschen Pendant (`/join #gentoo.de`) etwas ruhiger zugeht. Den Autor des Buches finden Sie im Kanal `#gentoo-web`.

IRC bietet zahlreiche Funktionen und Sie sollten sich auf jeden Fall die Dokumentation zu `irssi` genauer anschauen und ein wenig über die Netiquette im IRC informieren. Der Befehl `/help` in `irssi` ist schon einmal ein guter Einstieg.

16.8.3 Bugs einreichen

Früher oder später werden Sie bei dem ein oder anderen Paket auf einen Fehler stoßen. Je mehr instabile Pakete Sie verwenden, desto früher.

In diesem Fall *sollten* Sie einen Fehlerbericht unter `http://bugs.gentoo.org` einreichen. Sie sind in den seltensten Fällen der einzige Benutzer, der den Fehler zu spüren bekommt, und die Entwickler können nicht alle Situationen vorher testen. Darum lebt die Distribution von diesem Korrekturverfahren.

Die Bugzilla-Installation[10] führt recht komfortabel durch das Erstellen eines Fehlerberichts. Wenn möglich überprüft man vor Anlegen eines Bugs, ob dieses Problem bereits gemeldet wurde.

Der Fehlerbericht ist in Englisch zu verfassen, und Sie sollten darauf achten, dass auch Portage alle Fehlermeldungen in Englisch ausspuckt (vgl. Seite 189). Zum Abschluss hängen Sie die Informationen zu Ihrem System an, denn schließlich hat jeder Benutzer seine eigene Umgebung, die für das Problem eine Rolle spielen könnte. Grundsätzlich fügt man die Ausgabe des Befehls `emerge --info` (siehe auch Seite 127) an:

```
gentoo ~ # emerge --info
Portage 2.1.2.2 (default-linux/x86/2007.0, gcc-4.1.1, glibc-2.5-r0, 2.6.
19-gentoo-r5 i686)
=================================================================
System uname: 2.6.19-gentoo-r5 i686 Intel(R) Celeron(R) CPU 2.60GHz
Gentoo Base System release 1.12.9
Timestamp of tree: Thu, 08 Mar 2007 00:30:01 +0000
```

[9] Eine Liste finden Sie unter `http://www.gentoo.org/main/en/irc.xml`.

[10] `http://bugs.gentoo.org`

```
ccache version 2.4 [enabled]
dev-java/java-config: 1.3.7, 2.0.31
dev-lang/python:     2.4.3-r4
dev-python/pycrypto: 2.0.1-r5
dev-util/ccache:     2.4-r6
sys-apps/sandbox:    1.2.17
sys-devel/autoconf:  2.61
sys-devel/automake:  1.9.6-r2, 1.10
sys-devel/binutils:  2.16.1-r3
sys-devel/gcc-config: 1.3.14
sys-devel/libtool:   1.5.22
virtual/os-headers:  2.6.17-r2
ACCEPT_KEYWORDS="x86"
AUTOCLEAN="yes"
CBUILD="i686-pc-linux-gnu"
CFLAGS="-O2 -march=pentium4 -pipe -fomit-frame-pointer"
CHOST="i686-pc-linux-gnu"
CONFIG_PROTECT="/etc"
CONFIG_PROTECT_MASK="/etc/env.d /etc/env.d/java/ /etc/gconf /etc/java-co
nfig/vms/ /etc/php/apache1-php5/ext-active/ /etc/php/apache2-php5/ext-ac
tive/ /etc/php/cgi-php5/ext-active/ /etc/php/cli-php5/ext-active/ /etc/r
evdep-rebuild /etc/splash /etc/terminfo"
CXXFLAGS="-O2 -march=pentium4 -pipe -fomit-frame-pointer"
DISTDIR="/usr/portage/distfiles"
FEATURES="autoconfig ccache distlocks metadata-transfer parallel-fetch s
andbox sfperms strict"
GENTOO_MIRRORS="ftp://pandemonium.tiscali.de/pub/gentoo/ http://mirror.m
untinternet.net/pub/gentoo/ ftp://gentoo.imj.fr/pub/gentoo/ http://213.1
86.33.38/gentoo-distfiles/ ftp://ftp-stud.fht-esslingen.de/pub/Mirrors/g
entoo/"
LANG="de_DE.utf8"
LC_ALL="de_DE.utf8"
PKGDIR="/usr/portage/packages"
PORTAGE_RSYNC_OPTS="--recursive --links --safe-links --perms --times --c
ompress --force --whole-file --delete --delete-after --stats --timeout=1
80 --exclude=/distfiles --exclude=/local --exclude=/packages --filter=H_
**/files/digest-*"
PORTAGE_TMPDIR="/var/tmp"
PORTDIR="/usr/portage"
PORTDIR_OVERLAY="/usr/portage/local/overlay"
SYNC="rsync://rsync.europe.gentoo.org/gentoo-portage"
USE="acl apache2 berkdb bitmap-fonts cli cracklib crypt cups dri ...
Unset:  CTARGET, EMERGE_DEFAULT_OPTS, INSTALL_MASK, LDFLAGS, LINGUAS, MA
KEOPTS, PORTAGE_RSYNC_EXTRA_OPTS
```

Anhang

A Anhang

Die grafische Installation

Der grafische Installationsprozess verpackt zwar viele notwendige Installationsschritte recht hübsch, verlangt aber dennoch Gentoo-Grundwissen, das wir in diesem Buch in den Kapiteln 1 bis 6 vermitteln. Wenigstens das Installationskapitel 1 ab Seite 21 sollte bekannt sein, bevor man sich an die grafische Installation macht.

A.1 Booten der LiveDVD

Diesmal starten wir das System von der beiliegenden LiveDVD und betätigen bei der ersten Eingabeaufforderung (`boot:`) die Enter-Taste.

Wir gelangen so über die Abfrage der Keymap (siehe Kapitel 1.1 ab 23) und den Gentoo-Bootscreen in das grafische X-System. Sollte es Probleme geben, bevor der Bootscreen erscheint, wählt man den Kernel ohne Framebuffer-Unterstützung (`gentoo-nofb`). Die grafische Benutzeroberflä-

che sollte später trotzdem starten, aber man verzichtet auf den hübschen Bootscreen.

Sobald der X-Server läuft, loggt einen das System automatisch als User `gentoo` ein.

Leider informiert die Keymap-Abfrage während des Boot-Vorgangs unsere Benutzeroberfläche nicht darüber, dass wir eine deutsche Tastatur besitzen, und so müssen wir diesen Vorgang wiederholen. Die entsprechende Einstellung findet sich im Menü **System** bei den **Preferences** (**Einstellungen**) im Unterpunkt **Keyboard**. Hier lässt sich unter **Layouts** mit **Add** die deutsche Tastaturbelegung hinzufügen und als **Default** markieren. Haben wir das erledigt, müssen wir die Sonderzeichen nicht mehr erraten.

A.2 Der Gentoo Linux Installer

Der grafische Installer ist deutlich komfortabler als die Kommandozeile. Man sollte allerdings im Hinterkopf behalten, dass es sich bei dem Installer um eine recht junge Software handelt, die nicht jede Installation einwandfrei abschließen wird.

Bei Bedarf wechselt man mit der Tastenkombination `[Strg]` + `[Alt]` + `[F1]` jederzeit auf die Kommandozeile der LiveDVD. Vor allem bei Problemen sucht es sich hier leichter nach Ursachen. Die grafische Benutzeroberfläche erhält man mit der Kombination `[Strg]` + `[Alt]` + `[F7]` zurück.

Wir starten den Installer über einen Doppelklick auf das Icon **Gentoo Linux Installer (GTK+)**.

Installationsvariante

Der Installer lässt uns zunächst den gewünschten Installationstyp wählen. Wir beschreiben hier nur die Standard-Variante und wählen entsprechend **Standard**.

Die **Networkless**-Variante benötigt während der Installation keinen Zugriff auf das Netzwerk, und wir nutzen sie für eine lokale Installation, während **Advanced** dem Gentoo-Experten mehr Optionen bietet. Für diese Methode finden Sie die notwendigen Hintergrundinformationen in den ersten Kapiteln dieses Buches.

Mit **Next** gelangen wir zum nächsten Schritt der Installation.

Partitionierung, Mountpoints und Netzwerkverzeichnisse

Wir gelangen zur grafischen Partitionierung des Systems und orientieren uns an den Vorgaben des Installationskapitels (siehe Seite 27). Üblicher-

weise entfernen wir aktuell vorhandene Partitionen über **Clear partitions**, um anschließend neue Schritt für Schritt hinzufügen.

Im nachfolgenden Bildschirm sind mit **Add** die notwendigen Mount-Verzeichnisse festzulegen. Unter **Device** wählt man jeweils eine der zuvor angelegten Partitionen und setzt das gewünschte Mount-Verzeichnis. Die Swap-Partition benötigt keinen Mount-Punkt.

Schließlich lassen sich Netzwerk-NFS-Verzeichnisse in den Dateibaum einbinden. Dies ist jedoch keine Voraussetzung, und wir überspringen diesen Punkt hier.

Stage und Portage-Baum

Wir könnten nun eine Stage aus dem Internet herunterladen, doch nutzen wir zunächst einmal die Stage der DVD (**Build stage from files on Live-DVD**). Es geht schneller, und da wir das System später ohnehin aktualisieren, spielt die Stage keine große Rolle.

Zudem müssen wir im nächsten Schritt keine weiteren Angaben zur Herkunft des Portage-Baums machen, denn diesen übernimmt der Installer ebenfalls von der DVD.

Anschließend ist der Installer etwas länger beschäftigt und erstellt die Basis unseres neuen Gentoo-Systems.

Einstellungen zur make.conf

Im nächsten Schritt konfiguriert man die Datei `/etc/make.conf`. Das betrifft vor allem die Einstellung der `CFLAGS`, die wir genauer unter 1.6.1 ab Seite 38 beschrieben haben. Für die zur Bestimmung der korrekten `CFLAGS` notwendigen Prozessor-Informationen wechselt man mit `[Strg]` + `[Alt]` + `[F1]` auf die Kommandozeile und liest die Angaben wie auf Seite **??** beschrieben aus:

```
gentoo ~ # cat /proc/cpuinfo
```

Zurück mit `[Strg]` + `[Alt]` + `[F7]`.

Auf dieser Seite lassen sich auch die USE-Flags festlegen. Details dazu finden Sie unter 1.6.1 ab Seite 38. Hier sei jedoch empfohlen, die Standardeinstellung unverändert zu lassen und nur bei konkretem Bedarf später einzelne USE-Flags zu aktivieren.

Die unter **Other** gelisteten Einstellungen sollte man – bis auf die `MAKEOPTS` (siehe hierzu 6.3.1 ab Seite 149) – nicht verändern.

Vor allem raten wir davon ab, die kleine Checkbox **Use unstable (~arch)** zu verwenden. Mehr dazu unter 5.3.1 ab Seite 130.

Das root-Passwort

Nun setzen wir noch das `root`-Passwort, das wir zur Sicherheit zweimal eingeben.

Zeitzone

Wir wählen unseren Standort und können dazu sogar auf die Karte klicken, wobei es nicht ganz einfach ist, Berlin mit dem Mauszeiger zu treffen. Alternativ wählt man die gewünschte Stadt über die Scrollbox aus.

Kernel

Auch wenn als Standard **Build your own from sources** angewählt ist, machen wir es uns für eine Erstinstallation einfach; schließlich funktioniert der Kernel der LiveDVD für den eigenen Rechner einwandfrei, wenn man bis an diese Stelle der Installation gelangt ist. Also kopieren wir ihn einfach auf unser neues System hinüber, indem wir die Option **Use the kernel, initramfs, and modules from the LiveDVD** wählen. So erhalten wir erst einmal ein funktionsfähiges System und können später immer noch einen abgespeckten Kernel erstellen. Die nötige Anleitung dazu findet sich in Kapitel 2 ab Seite 61.

Netzwerk

War die Hardwareerkennung erfolgreich, sind nun die identifizierten Netzwerkkarten auszuwählen. Im einfachsten Fall wählt man das **eth0**-Interface, belässt die Konfiguration auf **DHCP** und speichert diese Karte über den **Save**-Knopf. Für kompliziertere Varianten ziehen Sie das Netzwerk-Kapitel 3 ab Seite 77 zu Rate.

In der Sektion **Hostname/Proxy information/Other** lässt sich derzeit nur der Host- und der Domain-Name eingeben. Dies sollte man tun, sofern man über ein entsprechend konfiguriertes Netzwerk verfügt.

Logger und cron-Daemon

Auf den nächsten beiden Seiten geht es um die bevorzugte `syslog`- und `cron`-Software, die der Installer für das System bereitstellt.

Anschließend werden einige weitere Werkzeuge für das Dateisystem automatisch installiert, bevor es schließlich zur Auswahl des Boot-Loaders geht.

Der Boot-Loader

Wir bleiben hier bei der Standard-Auswahl **grub**, das der Installer nach dem Klick auf **Next** installiert und konfiguriert. In diesem Schritt erstellt das Programm auch gleich den Boot-Sektor.

Benutzer, Pakete und Services

Im ersten der nächsten drei Schritte erlaubt der Installer das Anlegen von Benutzern, was aber nicht zwingend notwendig ist.

Über eine anschließende Paketauswahl lässt sich das System bereits weiter an die eigenen Bedürfnisse anpassen, es ist jedoch sinnvoller, zunächst die Installation abzuschließen und zu sehen, ob der Rechner auch einwandfrei startet. Weitere Pakete lassen sich jederzeit ergänzen.

Zuletzt bestimmen wir die zu startenden Services. Es empfiehlt sich, zusätzlich den `sshd`-Daemon anzuwählen, damit man sich nach dem erfolgreichen Start des Rechners auch per SSH einloggen kann.

/etc/conf.d/* und /etc/rc.conf

Es folgen einige notwendige Einstellungen für Uhrzeit, Tastatur und die grafische Benutzeroberfläche. Ohne diese Benutzeroberfläche bleiben **Display Manager** und **XSession** unverändert.

Bei der Einstellung **Clock** fehlt – wie häufig unter Gentoo – der Hinweis, dass bei gleichzeitiger Verwendung von Windows auf dem Rechner die korrekte Einstellung **local** lautet. Bei einem reinen Linux-Rechner sollte man allerdings **UTC** bevorzugen.

Bei der Wahl des Editors ist darauf zu achten, dass dieser auch tatsächlich installiert ist. Standardmäßig steht nur `nano` zur Verfügung.

Die **Windowkeys** setzen wir, wie schon in der Einleitung besprochen, auf **No**, als **Console Font** wählen wir **lat9-16** und als **Keymap** die **de-latin1-nodeadkeys**. Die **Extended Keymaps** bleiben ungesetzt.

Finale

Der Installer sollte sich daraufhin mit **Your install is complete!** verabschieden. Wir starten nun den Rechner neu, nehmen die DVD aus dem Laufwerk und prüfen, ob der Rechner problemlos hochfährt.

B Anhang

Eine Maschine mit Windows teilen

Trotz der Fortschritte, die Linux in den letzten Jahren erlebt hat, bleibt es eine Tatsache, dass Windows das am weitesten verbreitete Betriebssystem ist. Entsprechend häufig wird es schon auf dem Rechner vorinstalliert und die Option, die Festplatte komplett zu löschen, nicht wirklich akzeptabel sein.

Wir wollen hier nur einen ganz kurzen Leitfaden liefern, wie man eine Windows-NTFS-Partition (Windows XP, Windows Vista, ...) verkleinert und so Platz für Gentoo schafft. Ausführlichere Informationen dazu finden sich auf den entsprechenden Seiten des deutschen[1] oder des englischen Wikis[2].

Eine Warnung vorweg: Es gibt keine Garantie, dass sich die Windows-Partition problemlos verkleinern lässt, und man geht ein hohes Risiko ein, die Daten der Windows-Installation vollständig zu verlieren. Ein Backup der entsprechenden Daten ist darum selbstverständlich.

[1] `http://de.gentoo-wiki.com/Dual_Boot`
[2] `http://gentoo-wiki.com/HOWTO_Dual_boot`

B.1 Windows verkleinern

Unter Windows Vista greift man auf die Werkzeuge des Betriebssystems zurück und verkleinert Partitionen über das Festplattenmanagement in der Systemsteuerung. Das ist auch der empfohlene Weg, denn Windows weiß am ehesten, wie es mit seinen Partitionen umzugehen hat.

Windows XP bietet diese Funktionalität leider nicht, und um hier die Windows-Partition zu verkleinern, verwenden wir das Programm `ntfsresize`, das ebenfalls auf der LiveDVD zur Verfügung steht. In den neueren Versionen kann dieses Werkzeug Windows-Partitionen auch ohne vorherige Defragmentierung der Festplatte verkleinern. Früher war dieser Schritt unumgänglich, um keine Daten auf der Partition zu verlieren.

Angenommen unsere Windows-Partition befindet sich auf der Festplatte, die sich über die Gerätedatei `/dev/hda` ansprechen lässt, und stellt dort die erste Partition `/dev/hda1` dar, so holen wir uns mit `ntfsresize` zunächst einmal einige wichtige Informationen über diese Partition:

```
livecd ~ # ntfsresize --info /dev/hda1
ntfsresize v1.13.1 (libntfs 9:0:0)
Device name        : /dev/hda1
NTFS volume version: 3.1
Cluster size       : 4096 bytes
Current volume size: 298594595328 bytes (298595 MB)
Current device size: 298594598912 bytes (298595 MB)
Checking filesystem consistency ...
100.00 percent completed
Accounting clusters ...
Space in use       : 28833 MB (9.7%)
Collecting resizing constraints ...
You might resize at 28832407552 bytes or 28833 MB (freeing 269762 MB).
Please make a test run using both the -n and -s options before real
resizing!
```

Im hier gezeigten Beispiel belegt die Partition ca. 300 GB, von denen aber nur ca. 30 GB in Benutzung sind. Wir wollen die Windows-Partition auf 40 GB verkleinern.

Wir starten `ntfsresize` in einem „Trockendurchlauf“, um zu sehen, ob alles problemlos funktionieren wird. Dass wir uns in einem Probelauf befinden, teilen wir `ntfsresize` über die Option `--no-action` (bzw. `-n`) mit; die gewünschte Größe spezifizieren wir mit `--size` (bzw. `-s`).

```
livecd ~ # ntfsresize --no-action --size 40G /dev/hda1
ntfsresize v1.13.1 (libntfs 9:0:0)
Device name        : /dev/hda1
NTFS volume version: 3.1
Cluster size       : 4096 bytes
```

```
Current volume size: 298594595328 bytes (298595 MB)
Current device size: 298594598912 bytes (298595 MB)
New volume size    : 39999996416 bytes (40000 MB)
Checking filesystem consistency ...
100.00 percent completed
Accounting clusters ...
Space in use       : 28833 MB (9.7%)
Collecting resizing constraints ...
Needed relocations : 3113915 (12755 MB)
Schedule chkdsk for NTFS consistency check at Windows boot time ...
Resetting $LogFile ... (this might take a while)
Relocating needed data ...
100.00 percent completed
Updating $BadClust file ...
Updating $Bitmap file ...
Updating Boot record ...
The read-only test run ended successfully.
```

Wenn hier alles glatt läuft, entfernen wir die `--no-action`-Option und wagen uns an die tatsächliche Verkleinerung. Während der Aktion sollten wir natürlich unter keinen Umständen unterbrechen. Da `ntfsresize` diesmal wirklich in Aktion tritt, werden wir nochmals gefragt, ob die Aktion auch wirklich durchgeführt werden soll:

```
livecd ~ # ntfsresize --size 40G /dev/hda1
...
WARNING: Every sanity check passed and only the dangerous operations
left.
Make sure that important data has been backed up! Power outage or
computer
rash may result major data loss!
Are you sure you want to proceed (y/[n])? y
...
Successfully resized NTFS on device '/dev/sda1'.
You can go on to shrink the device for example with Linux fdisk.
IMPORTANT: When recreating the partition, make sure that you
  1)  create it at the same disk sector (use sector as the unit!)
  2)  create it with the same partition type (usually 7, HPFS/NTFS)
  3)  do not make it smaller than the new NTFS filesystem size
  4)  set the bootable flag for the partition if it existed before
Otherwise you won't be able to access NTFS or can't boot from the
disk!
If you make a mistake and don't have a partition table backup then you
can recover the partition table by TestDisk or Parted's rescue mode.
```

Nach Abschluss der Verkleinerung erhalten wir Instruktionen, wie wir nun mit `fdisk` vorgehen müssen, um die Partitionsgröße anzupassen. Diese Aktion wird `ntfsresize` nicht automatisch übernehmen!

Wir starten also `fdisk`:

```
livecd ~ # fdisk /dev/hda
```

Erst einmal lassen wir uns das aktuelle Layout anzeigen und notieren Start- und End-Zylinder der zu verkleinernden Partition.

```
Command (m for help): p
[...]
   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1       36303   291596288    7  HPFS/NTFS
Partition 1 does not end on cylinder boundary.
/dev/hda2           36303       38913    20971360+   f  W95 Ext'd
(LBA)
/dev/hda5           36303       38913    20971329    b  W95 FAT32
```

In dem gezeigten Fall beginnt `/dev/hda1` bei Zylinder 1 und läuft bis Zylinder 36303. Wir sehen hier auch an dem Hinweis `Partition 1 does not end on cylinder boundary`, dass `ntfsresize` seine Arbeit erledigt hat. Wir löschen die erste Partition mit `d`, gefolgt von `1`, und legen sie gleich danach, aber diesmal verkleinert, wieder an. Dafür verwenden wir `n`, `p` für eine primäre Partition und wieder die `1`, da wir nochmals die erste Partition belegen. Wir werden nach dem Start-Zylinder gefragt und geben wieder die `1` an, gefolgt von der gewünschten Größe der Partition, hier `+40000M`:

```
First cylinder (1-38913, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-36302, default 36302): +40
000M
```

Bleibt noch der Partitionstyp mit der Kombination `t`, `1` und `7` für NTFS zu vergeben und die erste Partition mit `a` sowie `1` als Boot-Partition zu markieren (das ist nur für den Windows-Bootloader wirklich notwendig). Geben wir die Partitionstabelle zur Sicherheit noch einmal aus:

```
Command (m for help): p
[...]
   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        4864    39070048+   7  HPFS/NTFS
/dev/hda2           36303       38913    20971360+   f  W95 Ext'd
(LBA)
/dev/hda5           36303       38913    20971329    b  W95 FAT32
```

Das sieht soweit gut aus, und wir haben Platz zwischen den Zylindern 4864 und 36303 gewonnen – gute 260 GB. Wir könnten in dem neu entstandenen Zwischenraum die neuen Gentoo-Partitionen entsprechend der Anleitung in Kapitel 1 anlegen. Besser ist es aber, die neue Partitionstabelle zunächst einmal zu schreiben und zu sehen, ob Windows noch problemlos startet.

Sollte es nämlich zu Problemen kommen, kann man an dieser Stelle immer noch zurück und die Partitionstabelle wieder entsprechend der vorher

notierten Werte zurücksetzen. Zwar lässt sich die Aktion von `ntfsresize` nicht rückgängig machen, aber eventuelle Fehler an der Partitionstabelle lassen sich so korrigieren.

Also schreiben wir die neue Partitionstabelle mit `w` und starten unser System neu:

```
livecd ~ # reboot
```

Selbstverständlich ist bei einem Neustart die LiveDVD rechtzeitig aus dem Laufwerk zu entfernen.

Ist dieser Test erfolgreich, booten wir wieder mit der LiveDVD und fahren entsprechend den Installationsanweisungen in Kapitel 1 fort. Wir kehren später an diesen Punkt zurück, wenn es darum geht, den Boot-Sektor zu schreiben.

B.2 Der Boot-Sektor

Die notwendige Konfiguration des Bootloaders für ein Dual-Boot-System ist denkbar einfach. Wir müssen nur folgende Zeilen an die Datei `/boot/grub/grub.conf` anhängen:

```
title=Windows
rootnoverify (hd0,0)
makeactive
chainloader +1
```

So erscheint beim Start eine Auswahl u. a. für **Windows**, die das Betriebssystem startet.