>> Portage Benutzer Leitfaden

[Bitte Kapitel auswählen]

1. Zu Beginn

1.1 Portage auf den aktuellen Stand bringen

Nachdem Sie Gentoo Linux installiert und ein wenig mit ihm gearbeitet haben, ist es möglich, dass Sie Fehler in einigen Paketen entdecken oder Sie wollen einfach nur die neusten Software Pakete installieren oder Ihre momentanen Pakete aktualisieren. Um dies durchzuführen, müssen Sie sich unseren neuesten Portage Baum herunterladen, bzw. Ihre lokale Kopie auf den neusten Stand bringen. Wir haben verschiedene Anonyme rsync-Server, die Sie benutzen können um dies zu tun. Hier erfahren Sie nun, wie es geht.

Benutzen Sie folgendes Kommando um Ihren Portage Baum zu synchronisieren.

Befehlsauflistung 1

```
# emerge rsync
```

Warnung
Bitte beachten Sie, dass *emerge sync* automatisch die **--clean** Option aktiviert, das heisst alle Ihre Änderungen in /usr/portage werden wieder gelöscht. Um Ihre eigenen Ebuilds aus dem Portage Baum auszulagern, benutzen Sie die **PORTAGE_OVERLAY** Funktion.

Wenn Sie anfangen viele Ebuilds zu schreiben, Bugs zu fixen und mehr, möchten Sie vielleicht Gentoo Linux Developer werden. Wenden Sie sich für weitere Informationen an Daniel Robbins oder Seemant Kulleen.

1.2 Portage updaten

Bevor Sie unseren Portage Baum verwenden können, ist es wichtig, dass Sie Ihre Portage auf den aktuellen Stand bringen. Dies tun Sie mit folgendem Kommando:

Befehlsauflistung 2: Portage aktualisieren

```
Dies Zeigt an, welche Pakete aktualisiert werden sollen.
# emerge -up system
Dies aktualisiert die entsprechenden Pakete
# emerge -u system
```

Nun benutzen Sie die neueste Version von Portage. Ab jetzt können Sie unser ebuild System benutzen, um Ihre installierte Software auf den neuesten Stand zu bringen.

2. Einführung von emerge

2.1 emerge --pretend

Bevor Sie ein Paket installieren, ist es grundsätzlich eine gute Idee zu schauen, welche Abhängigkeiten benötigt und welche Pakete aktualisiert werden müssen. *emerge --pretend* oder *emerge -p* kann dies für sie tun.

Befehlsauflistung 3: Emerge benutzen

```
# emerge -p xchat

These are the packages that I would merge, in order.

Calculating dependencies......... done!
[ebuild   U] sys-libs/zlib-1.1.3-r2 to /
[ebuild   U] dev-libs/glib-1.2.10 to /
[ebuild N  ] media-libs/jpeg-6b-r2 to /
[ebuild N  ] x11-base/xfree-4.0.3-r3 to /
```

```
[ebuild N  ] x11-libs/gtk+-1.2.10-r1 to /
[ebuild N  ] media-libs/giflib-4.1.0-r3 to /
[ebuild N  ] media-libs/tiff-3.5.6_beta to /
[ebuild N  ] media-libs/imlib-1.9.10 to /
[ebuild N  ] net-irc/xchat-1.4.3 to /
```

In diesem bestimmten Fall werden wir **xchat** auf einer Maschine ohne X installieren. So hat *emerge --pretend* richtig festgestellt, dass zunächst etliche Abhängigkeiten erfüllt werden müssen. Besonders sys-libs/zlib und dev-libs/glib müssen aktualisiert werden und ausserdem müssen noch einige Ebuilds (natürlich auch x11-base/xfree) eingespielt werden.

2.2 USE und emerge

Oben habe ich *emerge --pretend* auf einem System ausgeführt, welches *gnome* nicht in der *USE* Variable in /etc/make.conf definiert hat. Das bedeutet, dass eine optionale GNOME Unterstützung, falls vorhanden, nicht aktiviert wird. Wie auch immer, xchat **hat** optionale GNOME Unterstützung. Nun lassen Sie uns einen Blick auf die Ausgabe von *emerge --pretend* werfen, nachdem ich *gnome* zur *USE* Umgebungsvariable in /etc/make.conf ergänzt habe:

Befehlsauflistung 4: Emerge in Kombination mit USE Flags

```
# emerge -p xchat

These are the packages that I would merge, in order.

Calculating dependencies.......................... done!
[ebuild N  ] media-libs/jpeg-6b-r2 to /
[ebuild N  ] gnome-base/libghttp-1.0.9 to /
[ebuild N  ] media-libs/audiofile-0.2.1 to /
[ebuild N  ] media-sound/esound-0.2.22-r2 to /
[ebuild N  ] gnome-base/gnome-env-1.0 to /
[ebuild N  ] gnome-base/libxml-1.8.11 to /
[ebuild N  ] gnome-base/ORBit-0.5.8 to /
[ebuild N  ] gnome-base/oaf-0.6.5 to /
[ebuild   U] dev-libs/glib-1.2.10 to /
[ebuild N  ] net-libs/libwww-5.3.2-r1 to /
[ebuild N  ] media-libs/giflib-4.1.0-r3 to /
[ebuild N  ] dev-util/guile-1.4-r3 to /
[ebuild   U] sys-libs/zlib-1.1.3-r2 to /
[ebuild N  ] x11-base/xfree-4.0.3-r3 to /
[ebuild N  ] x11-libs/gtk+-1.2.10-r1 to /
[ebuild N  ] media-libs/tiff-3.5.6_beta to /
[ebuild N  ] media-libs/imlib-1.9.10 to /
[ebuild N  ] gnome-base/gnome-libs-1.2.13 to /
[ebuild N  ] gnome-base/glibwww-0.2-r1 to /
[ebuild N  ] gnome-base/gdk-pixbuf-0.11.0 to /
[ebuild N  ] gnome-base/gconf-1.0.0 to /
[ebuild N  ] gnome-base/gnome-vfs-1.0.1 to /
[ebuild N  ] gnome-base/control-center-1.4.0.1 to /
[ebuild N  ] gnome-base/scrollkeeper-0.2 to /
[ebuild N  ] dev-util/xml-i18n-tools-0.8.1 to /
[ebuild N  ] gnome-base/libglade-0.16-r1 to /
[ebuild N  ] gnome-base/gnome-core-1.4.0.4 to /
[ebuild N  ] net-irc/xchat-1.4.3 to /
```

Wie Sie sehen erkennt emerge Dank der gesetzten USE Variable, dass xchat optionale GNOME Unterstützung beinhalten sollte. Damit diese optionale GNOME Unterstützung korrekt kompiliert werden kann, muss zunächst GNOME installiert werden. *emerge* erkennt dies und ergänzt verschiedene Pakete, die zur GNOME Installation benötigt werden zur Liste der zu installierenden Pakete. In einigen Fällen kann es vorkommen, dass Ihre USE Variable nicht richtig gesetzt ist. Dadurch kann es passieren, dass *emerge* unerwartete optionale Erweiterungen aktiviert oder ausschliesst. Deshalb empfehle ich, dass Sie vor dem eigentlichen *emerge* immer ein *emerge --pretend* ausführen. Dies gilt besonders für neue oder ungewöhnliche Ebuilds. Dadurch wissen Sie, was Sie erwartet. Sobald alles OK aussieht können Sie mit dem eigentlich *emerge* durch weglassen der *--pretend* Option fortfahren:

Befehlsauflistung 5: xchat emergen

```
# emerge xchat
```

Nachdem alle Abhängigkeiten installiert sind (soweit es welche gibt; nicht alle Pakete haben welche) wird der xchat Source nach /usr/portage/distfiles runtergeladen, verifiziert (md5), entpackt, compiliert und in ein temporäres Verzeichnis Namens "Sandbox" installiert. Danach werden sie ins lokale Filesystem integriert und eine Paketdatenbank in /var/db/pkg/net-irc/xchat-1.4.3/CONTENTS erstellt. Diese Datenbank enthält die installierten Dateien und ihre md5-Checksummen.

Um zu sehen, welche USE Flags in Zusammenhang mit einem bestimmten Paket benutzt werden können fügen Sie das *-v* oder *--verbose* Argument an *emerge -p* an:

Befehlsauflistung 6: Benutzen von emerge mit --verbose

```
# emerge -pv gentoo-sources

These are the packages that I would merge, in order:

Calculating dependencies ...done!
[ebuild    U ] sys-kernel/gentoo-sources-2.4.20-r5 -build +crypt -evms2
-aavm -usagi
```

2.3 Herausfinden, was sich geändert hat

Wenn Sie herausfinden möchten, was sich zwischen der bereits installierten und der neueren Version eines Pakets geändert hat fügen Sie *--changelog* odr *-l* als Argument an das *emerge* Kommando an:

Befehlsauflistung 7: Anzeigen des ChangeLog

```
# emerge -pl mozilla

These are the packages that I would merge, in order:

Calculating dependencies ...done!
[ebuild    U ] net-www/mozilla-1.3-r1 [1.2.1-r5]

*mozilla-1.3-r1

       22 Mar 2003; Martin Schlemmer <azarah@gentoo.org> mozilla-1.3-r1.ebuild :
  Add Gtk2 patch.  Add default/prefs/xft.js when Xft is enabled.  Some other
  long overdue cleanups.

  *mozilla-1.3

  21 Mar 2003; Jay Kwak <jayskwak@gentoo.org> mozilla-1.3.ebuild :
  Add XIM input patch for GTK

  18 Mar 2003; Martin Schlemmer <azarah@gentoo.org> mozilla-1.3.ebuild :
  New version.

  13 Mar 2003; Olivier Reisch <doctomoe@gentoo.org> mozilla-1.2.1-r5.ebuild :
  Marked ppc stable

  *mozilla-1.3_beta

  23 Feb 2003; Martin Schlemmer <azarah@gentoo.org> mozilla-1.3_beta.ebuild :
  New version.
```

3. Pakete upgraden

Der Standardweg um Pakete unter Gentoo Linux zu aktualisieren ist *emerge --update* oder *emerge -u*.

Befehlsauflistung 8: Benutzen von emerge -u

```
# emerge -u xchat
```

Portage benutzt ein sogenanntes "sicheres" deinstallieren; es "löscht" nur Original Dateien. Wenn eine Datei überschrieben oder modifiziert wurde, bleibt diese im Dateisystem erhalten (vorrausgesetzt, dass Sie eine neuere Version installiert haben). Wenn Sie also eine alte Version von xchat nach der Installation einer neuen Version löschen (unmergen), wird die ausführbare Datei von xchat nicht gelöscht, solange sie einen neueren Zeitstempel bzw. eine andere md5-Checksumme hat. Dieses sichere Löschen ist grossartig, da es sicherstellt, dass jeder Zeit eine Version der Anwendung verfügbar ist. Würden Sie zuerst löschen, würde xchat für einige Minuten nicht verfügbar sein, bis die neue Version heruntergeladen, kompiliert und installiert wäre.

Wichtig
Portage hat nun eine spezielle Funktion, die "config file protection", eine Schutzfunktion für Konfigurationsdateien. Die Aufgabe dieser Funktion ist es zu verhindern, dass neue Pakete bereits existierende Konfiguration verändern. Schutz der Konfigurationsdateien ist automatisch für /etc und die KDE Konfigurationsverzeichnisse aktiviert. Um weiter Informationen dazu zu erhalten geben sie *emerge --help config* ein.

4. Auflösen von blockenden Paketen

Aktuell installierte Pakete können in machen Fällen die Installation anderer Pakete blockieren. Dies kann passieren, wenn die Funktionalität eines Pakets in ein anderes verschoben wurde oder zwei Pakete zueinander inkompatibel sind. Ein blockendes Paket muss zunächst deinstalliert werden, bevor das geblockte Paket installiert werden kann.

Befehlsauflistung 9: Emergen eines Pakets, dass geblockt ist

```
# emerge -pv libbonobo

These are the packages that I would merge, in order:

Calculating dependencies ...done!
[blocks B     ] gnome-base/bonobo-activation (from pkg gnome-base/libbonobo-2.4.0)
[ebuild     U ] gnome-base/ORBit2-2.8.1 [2.6.3] -doc +ssl
[ebuild     U ] gnome-base/libbonobo-2.4.0 [2.2.3] -doc
```

In dem oben dargestellten Beispiel blockt bonobo-activation die Installation von libbonobo-2.4.0.

Befehlsauflistung 10: Entfernen eines blockierenden Pakets aus dem System

```
# emerge -C bonobo-activation
# emerge libbonobo
```

Das Entfernen von bonobo-activation aus dem System erlaubt Ihnen libbonobo-2.4.0 erfolgreich zu installieren.

Wichtig
Das Ausführen von unmerge (*emerge -C*) entfernt ein Paket auch dann, wenn es als Abhängigkeit für andere Pakete benötigt wird. Dies kann Ihr System in einen Status bringen, in dem einige Programme oder gar das ganze System nicht mehr funktionieren.

5. Das Verhalten von Portage kontrollieren

Wenn Sie Anpassungen am Verhalten von Portage vornehmen möchten, sollten Sie /etc/make.conf editieren. Diese Datei enthält Variablen (oder Beispiele für Variablen), mit denen Sie das Verhalten von Portage ändern können. Wenn Sie zum Beispiel ändern möchten, wie Portage Sourcecode herunterlädt sollten Sie *FETCHCOMMAND* an Ihre Bedürfnisse anpassen.

/etc/make.conf enthält eine Vielzahl von Beispielen für Variableneinstellungen aus denen Sie die für Sie optimalen ableiten können. Sie sollten auch einen Blick in die make.conf Manpage (*man make.conf*) werfen und/oder das Portage Handbuch lesen.

Wenn Sie eine Variable nur für einen einzigen Lauf benötigen, können Sie sie anstelle des Editierens der /etc/make.conf als Umgebungsvariable setzen. *AUTOCLEAN="no" emerge kde* zum Beispiel deaktiviert das autocleaning während *emerge kde*.

6. Was sind maskierte Pakete?

Viele Leute sind neugierig, warum ein neu veröffentlichtes Paket nicht in einem *emerge -u world* enthalten ist. Ein gutes Beispiel ist xfree-4.3.0 (aktuell als diese Anleitung geschrieben wurde). Wenn Sie *emerge sync* gefolgt von einem *emerge -u world* ausführen werden Sie feststellen, das xfree nicht aktualisiert werden soll. Warum das?

Der Grund ist, dass bestimmte Pakete als "masked" markiert sind. Das heisst, dieses Paket wird nicht automatisch aktualisiert, solange Sie nicht bestimmte Aktionen ausführen um dies dennoch zu erreichen. Für eine Erklärung, wie die Installation von maskierten Pakete funktioniert, schauen Sie sich den Maskierte Pakete FAQ Thread in unseren Gentoo Foren an.

>> Portage Handbuch

[Bitte Kapitel auswählen]

1. Portage: Ein Überblick

1.1 Überblick

Portage ist ein sehr mächtiges und fortgeschrittenes Paket-Management-System. Seine Flexibilität und Fähigkeit als einfaches Werkzeug zum Kompilieren von Software oder als Herzstück einer brandaktuellen Linux Distribution zu dienen, ist nahezu einzigartig. Die Gentoo Linux Distribution wurde um Portage herum entwickelt.

Gentoo Linux wird oftmals als "Meta-Distribution" bezeichnet. Gentoo besteht aus Portage und über 4000 Anleitungen zum Kompilieren von Paketen, sogenannten **ebuilds**. Diese **ebuilds** geben Portage die Anweisungen wie ein bestimmtes Softwarepaket kompiliert und installiert werden soll. Durch die Benutzung von **Profilen** und dem Kommandozeilen Programm **emerge** können Benutzer und Entwickler Portage dazu nutzen um die Pakete zu installieren und zu pflegen, die die Grundlage des Betriebssystems und der darauf laufenden Anwendungen darstellen.

Ein Gentoo Linux wird "on-the-fly" kompiliert, d.h. direkt auf den entsprechenden Rechner angepasst und erstellt. Der Installationsprozess umfasst das Erstellen eines funktionierenden Compilers sowie einer minimalen Umgebung, in der Portage Quellcode aus dem Internet laden kann, um den Rest des "Systemkerns" und etwaige Anwendungen zu installieren. Auch wenn Portage die Benutzung von vorkompilierten Anwendungen unterstützt, stellt diese nur einen Kompromiss dar und wird nur zur Installation auf langsamen Maschinen bzw. von Entwicklern die schnell ein bestimmtes Paket wiederherstellen wollen verwendet. Desweiteren gibt es einem die Möglichkeit, Pakete auf einer schnellen Maschine zu kompilieren, um sie dann auf einem langsamen Rechner zu installieren.

Durch das "on-the-fly" Kompilieren und die starke Konfigurierbarkeit von Portage, sind nur sehr wenige Gentoo Installationen gleich. Im Grunde wird bei der Installation von Gentoo Linux eine angepasste Linux Distribution erstellt, die sich an den Optionen, wie sie in der Portage Konfiguration und den ebuilds selber definiert sind, orientiert.

Auf den ersten Blick mag die Idee hinter Portage ähnlich dem BSD Ports System sein. Beide kompilieren Pakete aus den Sourcen und erlauben dem Benutzer Software sicher zu installieren bzw. zu deinstallieren. Und beide lösen Abhängigkeiten automatisch auf. Viele Ideen von Portage wurden beim BSD Ports System ausgeliehen, jedoch handelt es sich bei Portage definitiv nicht um eine weitere "Ports-Kopie".

Das Portage System ist eine Verbindung aus einem auf Python basierenden Kern und auf Bash Scripten basierten **ebuilds**. Anstatt mit Makefiles und dem *make* Kommando zu hantieren, verbindet Portage die Möglichkeiten von Python und das einfach zu handhabende Shellscripting mit einigen objektbasierten Eigenschaften, um ein einzigartig mächtiges System zu erstellen. Dieses setzt Portage an die Spitze aller aktuellen Ports Systeme.

Einige der von Portage angebotenen erweiterten Funktionen sind die Möglichkeit verschiedene Versionen und Überarbeitungen des gleichen Pakets zu halten, Auflösung der entsprechenden Abhängigkeiten, feinstrukturiertes Paketmanagement, sichere Installation via Sandbox, Schutz von bestehenden Konfigurationsdateien, Profile und vieles mehr. Viele dieser Funktionen werden noch im Laufe dieses Handbuches genauer erläutert.

1.2 Umgebungsabhängige Auflösung von Abhängigkeiten und Unterstützung von Features

Das Portage System ist im Hinblick auf die dem Benutzer gebotene Flexibilität einzigartig. Traditionelle BSD Ports Systeme tendieren dazu nur jeweils eine Version eines Paketes zu installieren. Portage hat diese Begrenzungen nicht. Es können mehrere Versionen eines Paketes installiert werden. Paketabhängigkeiten (Pakete, die zum Kompilieren andere Pakete nötig sind) können entweder mit ihrem Namen oder mittels zusätzlich angehangener Versionsnummer definiert werden. Das ermöglicht mehrere Versionen eines Paketes zeitgleich zu pflegen.

Das Abhängigkeits-System unterstützt ebenfalls umgebungbedingte Abhängigkeiten. Portage hat dafür ein leistungsstarkes Konzept, die sogenannten **USE Settings**. Durch ändern einer Konfigurations-Variablen in einer Portage Konfiguration, ist es möglich bestimmte Unterstützung für Funktionen oder

Bibliotheken während des Kompilierens zu aktivieren bzw. abzuschalten. Dies ist ein sehr flexibles und leistungsstarkes System, welchem wir uns im nächsten Kapitel widmen werden.

Zusätzlich unterstützt Portage Konzept der **SLOTs**. Während der Entwicklung von Gentoo Linux wurde klar, dass wir öfters mehrere Versionen bestimmter Pakete (wie z.B. Bibliotheken) benötigen, um die Ansprüche andere Pakete zu erfüllen. Der traditionelle Weg um dieses Problem zu lösen war, verschiedene Versionen eines Paketes als unterschiedliche Pakete mit leicht abweichenden Namen zu behandeln.

Anstatt verschiedene Versionen als eigene Pakete zu behandeln, brachten die Entwickler Portage bei, wie mehrere Versionen eines Paketes gleichzeitig durch Benutztung von **SLOTs** zu handhaben sind. Als Beispiel sei hier die gängige Freetype Bibliothek genannt. Die 1.x Reihe von Freetype ist mit der 2.x Reihe inkompatibel, jedoch sind oftmals beide Versionen nötig, um die Abhängigkeiten verschiedener Pakete zu erfüllen. Die meisten Distributionen und Ports Systeme tendieren dazu, ein "freetype" Paket für Freetype 1.x und ein "freetype2" Paket für Freetype 2.x anzubieten. Dies betrachten wir als beinahe komplett beschädigtes Paketmanagement System. Wir ergänzen einfach die *SLOT* Nummer 1 zur ersten und Nummer 2 zur zweiten Version. Mit dieser Information ist Portage in der Lage beide Zweige zu pflegen und, falls nötig, beide auf höhere Versionen zu updaten.

1.3 Profile

Portage unterstützt ein weiteres Konzept, sogenannte **Profile**. Ein Profil enthält eine Liste von Paketnamen und Versionen mit Anweisungen und einigen Standardoptionen, welche von Portage benutzt werden sollen. Ein Profil enthält Informationen welche Pakete und welche Versionen jeweils erlaubt bzw. verboten oder als zwingend notwendig behandelt werden sollen. Der Benutzer kann zwischen verschiedenen Profilen wechseln in dem er einfach einen Symlink ändert (/etc/make.profile). Dieses erscheint einfach, aber es erlaubt Portage den Kern einer Distribution anzupassen und als hochwertiges Build System zu dienen.

Der gesamte Aufwand, der bei der Entwicklung von Gentoo Linux betrieben wurde, resultiert in einer Sammlung von **ebuilds** und einem Profil. Dieses Profil beschreibt welche Pakete als Kernpakete für den Betrieb des Systems wichtig sind. Das Profil erlaubt darüber hinaus den Entwicklern bestimmte Pakete zu blockieren bzw. freizuschalten. Dies ist auch nützlich, um defekte Pakete temporär zu deaktivieren. Die **ebuild** Dateien definieren einfach nur wie bestimmte Pakete von Portage kompiliert und installiert werden sollen, die durch das Profil verlangt bzw. erlaubt werden.

2. Portage konfigurieren

2.1 Überblick

Dieses Kapitel soll verschiedene Aspekte der Konfiguration von Portage abdecken, die für Benutzer wie Entwickler wichtig sind. Portage ist ein sehr flexibles System und Sie müssen verstehen, wie man Portage richtig konfiguriert, um Ihr System genau auf Ihre Bedürfnisse abstimmen zu können.

Bitte beachten Sie, dass in diesem Dokument der Begriff "Benutzer" Personen mit administrativen Rechten definiert. Um die Portage Konfiguration zu ändern und neue Pakete zu installieren bzw. zu entfernen benötigen Sie Root-Rechte.

2.2 Portage Konfigurations Dateien

Fast alle Konfigurations-Optionen, die weiter unten erklärt werden, befinden sich in den Dateien /etc/make.conf /etc/make.profile/make.defaults und /etc/make.globals. /etc/make.conf enthält einige Optionen, die von Portage benutzt werden. Portage wird erst aktuell gesetzte Umgebungsvariablen für jegliche Optionen überprüfen. Werden keine Umgebungsvariablen gefunden, wird die Datei /etc/make.conf überprüft. Können dort auch keine Optionen gefunden werden, wird /etc/make.profile/make.defaults überprüft. Ist dort ebenso keine Option vorhanden, werden die Standardwerte aus der /etc/make.globals verwendet. Beachten Sie, dass alle Optionen, die in /etc/make.conf definiert sind, von den Optionen, die in /etc/make.globals gemacht wurden, fast immer überschrieben werden. Alle Optionen in /etc/make.conf und /etc/make.globals können systemweit als global betrachtet werden, solang es Portage betrifft.

Wenn Sie überprüfen wollen, ob eine Option bereits definiert wurde, ist es zu empfehlen, dass Sie zuerst /etc/make.conf überprüfen und erst dannach /etc/make.globals. Sofern nicht anders angegeben, werden Optionen, die in /etc/make.globals definiert wurden, von den Optionen in /etc/make.conf überschrieben.

2.3 USE Einstellungen

Das USE-System ist eine flexible Möglichkeit, um sämtliche Features, direkt beim Kompilierprozess von Gentoo Linux oder später für einzelne Pakete ein- bzw. auszuschalten. Dies erlaubt eine Administrierbarkeit, wie die Pakete in Verbindung mit den angegebenen Features in der USE-Variable, kompiliert werden. Zum Beispiel, wenn ein Paket Unterstützung für GNOME bietet, können Sie diese abschalten, indem Sie in der USE-Variable *-gnome* angeben. Wenn Sie jedoch die GNOME-Unterstützung für das Paket aktivieren wollen, dann setzen Sie in der USE-Variable *gnome* ein.

Der Effekt der USE Flags auf Pakete ist abhängig davon, ob die Software und das ebuild Paket selbst, das jeweils gesetzte USE Flag unterstützten. Wenn dem nicht so ist, so bewirkt auch das jeweilige USE Flag nichts an der Software. Auch werden einige Paketabhängigkeiten bei mancher Software nicht als optional angesehen, dadurch haben gesetzte USE Flags keine Auswirkung auf diese vorgeschriebenen Abhängigkeiten. Eine Liste der USE Flags, die von einem Paket benutzt werden, kann durch *emerge --verbose --pretend ebuild* herausgefunden werden.

Eine Liste der USE Flags, die von Gentoo Linux benutzt werden, können Sie aus der Datei /usr/portage/profiles/use.desc beziehen. Jedes USE Flag ist jeweils in einer Zeile mit einer zusätzlichen Beschreibung erklärt, welches Feature es aktiviert.

Portage bestimmt, welche USE Flags ein- bzw. ausgeschaltet sind aufgrund einer Überprüfung von bis zu vier verschiedenen Stellen, wo USE definiert sein kann. Diese Stellen bewirken, dass die USE Flags, die dort definiert sind, "aufgestapelt" werden. Es durchforstet jede einzelne Stelle und merkt sich jedes USE Flag der früheren Stellen, ob sie ein- bzw. ausgeschaltet sind und fügt neu gefundene Features hinzu, so dass es eine lange USE Variable ergibt. Wird zum Beispiel ein Feature an einer früheren Stelle abgeschaltet und an der aktuellen Stelle, die Portage durchforstet, eingeschaltet, so wird das Feature trotz Abschaltens an der früheren Stelle wieder eingeschaltet.

Portage überprüft die Stellen in denen USE Flags definiert werden in der Reihenfolge, wie sie in der Variable **USE_ORDER** in /etc/make.globals angegeben ist. Möchten Sie eine Stelle ausschalten, dann entfernen Sie diese einfach in **USE_ORDER**.

Die folgenden Absätze beschreiben jeden Ort, der in **USE_ORDER** mit der Portage Standard Konfiguration definiert wurde.

Defaults

Portage Profile können einen Satz von Standard USE Flags beinhalten. Diese Standard Features sind bei jedem Profil dabei und werden in der Datei make.defaults angegeben. Da /etc/make.profile als eine Verknüpfung zum aktuell benutzten Portage Profil fungiert, ist der Satz an standard USE Flags des aktuellen Profils die Datei /etc/make.profile/make.defaults. Es ist nicht zu empfehlen Änderungen in dieser Datei vorzunehmen, da zukünftige Profiländerungen sie wieder überschreiben würden.

Auto

Diese Pakete sind in der Datei use.defaults im Portage Profil (/etc/make.profile/use.defaults) definiert. Jeder Eintrag beinhaltet eine USE Flag Bezeichnung und das dazugehörende Paket. Wenn das für ein USE Flag angegebene Paket installiert ist, wird das Feature als eingeschaltet betrachtet. Wenn Sie zum Beispiel das Paket *x11-base/xfree* installieren und das USE Flag X nicht abschalten, so wird diese solange global eingeschaltet, wie das Paket installiert ist. Es ist nicht zu empfehlen, diese Datei zu bearbeiten da zukünftige Profiländerungen, diese überschreiben würden.

Conf

Wenn eine *USE* Variable in /etc/make.conf definiert ist, werden die *USE* Flags, die in jener Variable angegeben wurden, verwendet. Wenn es keine USE Variable in make.conf gibt, dann wird /etc/make.globals überprüft. Solch ein Eintrag könnte wie folgt aussehen:

Befehlsauflistung 1: Beispielhafte USE Variable

```
USE="slang readline gpm berkdb gdbm tcpd pam libwww ssl gb tk
lm_sensors lvm ldap tex bonobo sdl gtk xfs evo pda ldap
mmx mitshm perl python guile ruby postgres dvd 3dnow tcl
lcms gif sdl vorbis ogg oss libg++ directfb decss snmp
gnome X opengl mozilla pdflib gpg -nls gd xface jpilot
-kde -qt -esd -motif -alsa oggvorbis"
```

USE Flags werden eingeschaltet, indem Sie einfach in der USE Variable angegeben werden. USE Flags können jedoch auch mittels Negation durch ein - (Minus) vor dem Feature-Namen deaktiviert werden. Zum Beispiel schaltet *gnome* GNOME Support ein, während *-motif* den MOTIF Support abschaltet.

Die *USE* Variable in /etc/make.conf ist der empfohlene Ort, wo Sie USE Flags ein- bzw. ausschalten. Diese Datei wird nicht automatisch von Portage überschrieben. Es wird empfohlen, dass Sie hier die USE Flags eintragen, die nicht von den vorangehenden Stellen ein bzw. ausgeschaltet werden sollen.

Env

Die USE Flags kann man von Hand aus mit seiner eigenen Shell Umgebungsvariable überschreiben.

Befehlsauflistung 2: USE Flags temporär setzen

```
# export USE="-gnome"
# emerge net-im/gaim

oder

# USE="-gnome" emerge net-im/gaim
```

Dies erlaubt die Benutzung von speziellen USE Flags für ein oder mehrere Pakete. Diese von Ihnen per Hand definierten USE Flags bleiben bestehen, solange Sie in jener Shell Umgebung damit arbeiten. Verlassen Sie Ihre Umgebung ("exit"), ist auch das von Hand gesetzte USE Flag weg.

Notiz
Auch wenn Portage die aktuellen USE Flags zum Zeitpunkt der Installation des Pakets speichert, so sind diese Einstellungen nicht dauerhaft gespeichert. Sollte dieses Paket in der Zukunft erneut installiert bzw. aktualisiert werden, werden nur die aktuell im System definierten USE Flags verwendet und keinesfalls die zum Zeitpunkt der Erstinstallation geltenden.

2.4 Compiler Optionen

Die Compiler Optionen, welche von Portage zum kompilieren von Paketen genutzt werden, können in /etc/make.conf gesetzt werden indem *CHOST*, *CFLAGS* und *CXXFLAGS* editiert werden. Die *CHOST* Einstellung gibt an, auf welcher Plattform Sie kompilieren. *CFLAGS* und *CXXFLAGS* geben Compiler Einstellungen für C bzw. C++ an.

Es werden Ihnen einige kommentierte Standard Einstellungen für unterschiedliche Plattformen angeboten. Diese Einstellungen können als getestet und stabil angesehen werden. Diese zu ändern, könnte zu Fehlern in der Software und dem Compiler führen. Bitte seien Sie beim Ändern der Standard Einstellung vorsichtig, da sie zu einem unbenutzbaren bzw. beschädigten System führen können.

Benutzer eines Multiprozessor Systems können einige Vorteile erhalten, indem sie in /etc/make.globals die Option *MAKEOPTS* editieren. Diese Option wird während des Kompilierungsvorganges an *make* übergeben um mehrere *gcc* Instanzen zu aktivieren.

2.5 Lage der Verzeichnisse

Portage erlaubt es Ihnen zu bestimmen in welchen Verzeichnissen Pakete kompiliert und verschiedene Dateien gespeichert werden sollen. Die meisten Benutzer werden diese Optionen nicht benötigen. Folgende Optionen können genutzt werden:

- *PORTDIR* - Position des Portage Trees
- *DISTDIR* - Lokaler Cache der runtergeladenen Pakete
- *PKGDIR* - Ort der lokal erzeugten tbz2 Pakete
- *RPMDIR* - Ort der lokal erzeugten RPM Pakete
- *PORTAGE_TMPDIR* - Temporärer Platz zum Kompilieren der Pakete
- *BUILD_PREFIX* - Gehört zu PORTAGE_TMPDIR
- *PKG_TMPDIR* - Gehört zu PKGDIR

2.6 Schutz der Konfigurationsdateien

Portage erlaubt den Schutz aller Konfigurationsdateien in bestimmten Verzeichnissen. Portage wird in geschützten Verzeichnissen keine Dateien überschreiben. Wenn ein Paket versucht, bereits existierende Dateien zu überschreiben, wird die neue Datei nach dem Muster ._cfg0000_name umbenannt. Dies geschieht damit der Benutzer später die neue Datei einsehen kann, um die Unterschiede manuell einzufügen.

Geschützte Verzeichnisse können über *CONFIG_PROTECT* entweder in /etc/make.conf oder /etc/make.globals definiert werden. Bestimmte Dateien und Unterverzeichnisse unterhalb geschützter Verzeichnisse können durch *CONFIG_PROTECT_MASK* vom Schutz ausgenommen werden.

Das Folgende ist nur ein Beispiel und sollte nicht blind kopiert werden:

Befehlsauflistung 3: Beispielhafte Angabe zum Schutz von Konfigurationsdateien

```
CONFIG_PROTECT="/etc /usr/share/config /usr/kde/2/share/config"
CONFIG_PROTECT_MASK="/etc/gconf /etc/init.d /etc/pam.d"
```

Benutzer können mehr Informationen über den Schutz von Konfigurationsdateien erhalten, indem sie folgendes in ihrer Shell eingeben:

Befehlsauflistung 4: Hilfe zum Schutz von Konfigurationsdateien aufrufen

```
# emerge --help config
```

2.7 FEATURES

Portage bietet eine ganze Menge Optionen, die auf Entwickler abzielen, welche regeln wollen wie sich Portage verhält und nach der Installation verschiedener Paketstufen aufräumt. Diese Optionen sollten nur für Entwickler interessant sein. Bei Benutzern könnten sie Probleme verursachen.

Eine Liste einzuschaltender Funktionen kann unter **FEATURES** entweder in /etc/make.conf oder /etc/make.globals gesetzt. Hier eine Liste der verfügbaren Funktionen:

- *digest* : Erstellt automatisch einen Digest für neue Ebuilds.
- *cvs* : Ergänzt neue Ebuild Digests automatisch ins CVS.
- *sandbox* : Aktiviert das Sandboxverfahren.
- *noclean* : Portage räumt nach dem kompilieren nicht auf.
- *noauto* : Führt keine wichtigen Ebuild Schritte automatisch aus.
- *distcc* : Benutzt das verteilte Kompilieren mit distcc.
- *ccache* : Behält kompilierte Objektdateien, sodass eine erneute Kompilierung weniger Zeit benötigt.
- *buildpkg* : Erstellt Binärpakete für jedes Paket, das Sie emergen.
- *userpriv* : Kompiliert nicht mit root-Privilegien.
- *usersandbox* : Benutzt eine Sandbox, wenn **userpriv** aktiviert ist.
- *keeptemp* : Entfernt nicht die temporären Dateien (${T}) nach dem mergen.

Einige dieser Funktionen können durch Setzen der folgenden Variablen angepasst werden:

- *CCACHE_SIZE* : Damit wird festgelegt, wie viel Platz *ccache* benutzen kann. Standard sind 2GB.
- *DISTCC_HOSTS* : Legt fest, welche Rechner Teil des verteilten Kompilierens mit *distcc* sind. Alle diese Rechner müssen den *distccd* Daemon laufen haben.

2.8 Portage SYNC Einstellungen

Portage bietet die Möglichkeit den Portage Tree mittels*rsync* aufzufrischen. Finden Sie einen anderen Weg, können Sie angeben, welche Methode Portage verwenden soll, indem Sie ihn in der *SYNC* Option in der /etc/make.conf angeben.

RSYNC

rsync ist der gewöhnliche Weg Ihren Portage Tree aktuell zu halten. Die rsync Methode wird in der /etc/make.conf gesetzt:

Befehlsauflistung 5: Den Portage Tree mittels rsync aktuell halten

```
SYNC="rsync://rsync.gentoo.org/gentoo-portage"
```

Notiz
Die rsync-Methode überschreibt blind alle Veränderungen, die an Ihrem lokalen Portage Tree vorgenommen wurden. Falls Sie lokale Änderungen behalten möchten, müssen Sie *PORTDIR_OVERLAY="/ein/verzeichnis/fuer/Ihre/Aenderungen"* setzen.

Entwickler CVS

Entwickler mit vollem Zugang zum CVS können auch per **emerge** den lokalen Tree mit dem CVS abgleichen lassen, welches sie mittels SSH erreichen können.

Laden Sie einfach den CVS Tree mittels Ihres Accounts herunter ("checkout"), verschieben Sie diesen nach /usr/portage und benutzen Sie folgende **SYNC** Option:

Befehlsauflistung 6: Den Portage Tree mittels Entwickler-CVS aktuell halten

```
SYNC="cvs://Ihr_account@cvs.gentoo.org:/home/cvsroot"
```

2.9 Spiegelungen (Mirrors)

Das Gentoo Projekt bietet eine lokale Spiegelung von allen Quellpaketen, die in ebuilds im Portage Tree angegeben sind. Meistens werden Source Tar-Archive usw. auf langsamen Servern gespeichert, die ziemlich oft ausfallen können. Auch Entwickler entfernen regelmässig alte Versionen Ihrer Software von FTP-Servern, wenn neue Versionen herausgegeben werden. Um das Leben von Benutzer, die die Gentoo Distribution nutzen, zu erleichtern, spiegeln wir diese Dateien. Das erlaubt ein schnelleres und sicheres Herunterladen der Archivdateien von Spiegel-Servern, die näher an Ihrem Standort liegen.

Wann auch immer Sie ein Paket mergen und dazu das Quellpaket benötigen, versucht Portage zu allererst unseren Spiegel-Server für die angeforderten Dateien zu erreichen. Wenn sich auf diesem Server die benötigten Dateien nicht befinden, versucht Portage den jeweiligen HTTP- oder FTP-Server, der im ebuild angegeben ist.

Die Angabe des Spiegel-Server, den Portage benutzt, erfolgt in der *GENTOO_MIRRORS* Option, die sich in der Datei /etc/make.conf befindet. Das Folgende ist die standard Einstellung:

Befehlsauflistung 7: Download-Mirrors angeben

```
GENTOO_MIRRORS="http://www.ibiblio.org/gentoo"
```

Für einen Gentoo-Spiegel-Server in Ihrer Nähe, besuchen Sie die Gentoo Webseite oder fragen Sie in der lokalen Mailingliste nach einem für Sie günstigeren Server. Das Programm *mirrorselect* überprüft anhand von pings, welcher der für Sie am schnellsten erreichbare Server ist und fügt optional eine Liste von Servern in die GENTOO_MIRRORS Option in der /etc/make.conf ein. Schauen Sie in die Installationsanleitung für die Benutzung von *mirrorselect*.

2.10 Download-Programm

Das Programm, das Portage zum Download der benötigten Dateien verwendet, kann durch die Optionen *FETCHCOMMAND* und *RESUMECOMMAND* angegeben werden. Einige Beispiele werden in /etc/make.conf und /etc/make.globals gezeigt. Als Standardprogramm verwendet Portage *wget*, welches den meisten Anforderungen genügen sollte.

Notiz
Portage gibt Informationen über HTTP und FTP Proxies an das jeweilige Download-Programm weiter, die in den HTTP_PROXY und FTP_PROXY Optionen definiert sind.

2.11 Proxies

Portage kann angewiesen werden, zum Download HTTP oder FTP Proxies zu nutzen. Die Proxies können bei den Optionen *HTTP_PROXY* und *FTP_PROXY* entweder in /etc/make.conf oder in /etc/make.globals angegeben werden. Auch hier sollten die Einstellungen vorzugsweise in /etc/make.conf erfolgen. Wenn HTTP und FTP über den gleichen Proxy erfolgen soll, können Sie auch einfach die Option *PROXY* nutzen.

Im Folgenden ein Beispiel:

Befehlsauflistung 8: HTTP- und FTP-Proxies setzen

```
HTTP_PROXY="http://192.168.1.1:8080"
FTP_PROXY="http://192.168.1.1:8080"

oder

PROXY="http://192.168.1.1:8080"
```

Portage kann zusätzlich angewiesen werden einen HTTP Proxy zur Verwendung von rsync zu nutzen. Dies kann durch die Option *RSYNC_PROXY* in /etc/make.conf oder als Umgebungsvariable aktiviert werden.

Im Folgenden ein Beispiel:

Befehlsauflistung 9: RSYNC-Proxy setzen

```
RSYNC_PROXY="192.168.1.1:8080"
```

Notiz
Sollten Sie hinter einer Firewall sitzen und rsync scheint Ihren HTTP Proxy nicht nutzen zu können, können Sie Ihren Portage Tree durch einen Snapshot Tar-Archiv updaten. Schauen Sie in unsere FAQ für weitere Informationen.

2.12 Verschiedene Optionen

Die folgenden Optionen können für einige Nutzer sinnvoll sein:

- *NOCOLOR* : Erlaubt es dem Benutzer, das Verwenden von Farben bei Ausgaben von emerge zu deaktivieren.
- *CLEAN_DELAY* : Dadurch verzögert Portage das Entfernen von Paketen, um Nutzern die Möglichkeit zum Abbruch zu geben. Diese Zeit kann hier angegeben werden. Zum Abschalten der Verzögerung setzen Sie den Wert auf "0".

3. Paketmanagement

3.1 Updaten des Portage Trees

Der Portage Tree, der in /usr/portage liegt, enthält die Bibliothek der "Bauanleitungen" für verschiedene Pakete (sogenannte *ebuilds*). Darüber hinaus enthält der Tree auch noch Informationen zu dem Profil und der package.mask, die wichtig sind, um das System aktuell zu halten. Um immer die aktuellsten Versionen und neuesten Bugfixes zu haben, ist es wichtig den Tree regelmäßig mit dem offiziellen Tree abzugleichen. Sie können den Portage Tree mittels folgendem Kommando updaten:

Befehlsauflistung 10: Den Portage-Tree updaten

```
# emerge sync
```

Die von Portage genutzte Methode, kann geändert werden. Schauen Sie in die Portage SYNC Settings im Portage Konfigurieren Kapitel für weitere Informationen.

3.2 Pakete installieren (mergen)

Der Vorgang des Kompilierens und Installierens eines Paketes durch Portage wird als *mergen* bezeichnet. Portage kompiliert Pakete und installiert diese temporär in ein "Abbild-Verzeichnis", in dem es die zu installierenden Dateien aufzeichnet. Diese Dateien werden dann aus dem "Abbild-Verzeichnis" ins Root (/) Dateisystem integriert (gemerged).

Das *emerge* Kommando dient als Front-End des Portage Systems. Das Installieren und Entfernen von Paketen wird durch dieses Kommando und seine diversen Argumente kontrolliert.

Um die neueste, unmaskierte Version eines Paketes zu installieren, geben Sie einfach den Paketnamen, wie folgt ein (Beispiel):

Befehlsauflistung 11: Ein Paket installieren

```
# emerge galeon
```

Dieses Kommando wird zunächst alle benötigten Abhängigkeiten (unter Berücksichtigung der USE Flags) und dann die neueste und unmaskierte Version von Galeon kompilieren und installieren. Galeon könnte auch mit vollem Namen inklusive Kategorie angegeben werden: *net-www/galeon*

Das *emerge* Kommando akzeptiert auch die direkte Angabe von ebuilds. Dies erlaubt dem Benutzer auch ältere Versionen eines bestimmten Paketes oder ebuilds von Drittanbietern zu installieren. Beachten Sie, dass dadurch alle Maskierungen für das Paket übergangen werden und Ihre *ACCEPT_KEYWORDS* Einstellung ignoriert wird. Das Folgende ist ein Beispiel:

Befehlsauflistung 12: Ein Paket unter Angabe der Version installieren

```
# emerge /usr/portage/net-www/galeon/galeon-1.2.0-r3.ebuild
```

Zusätzlich zur Angabe des Paketnamen oder ebuild, unterstützt *emerge* verschiedene weitere Argumente. Eines dieser Argumente ist *--pretend*, vielleicht eines der nützlichsten. Durch dieses Argument wird das geplante Vorgehen nicht durchgeführt. Stattdessen gibt Portage eine Liste aller zu

installierenden Pakete aus. Das Folgende zeigt eine Auflistung der Pakete, die während der Installation der neuesten Version des Kdevelop Paketes installiert würden:

Befehlsauflistung 13: Darstellung der zu installierenden/aufzufrischenden Pakete

```
# emerge --pretend kdevelop

These are the packages that I would merge, in order.

Calculating dependencies ...done!
[ebuild  N   ] kde-base/kdelibs-2.2.2-r4 to /
[ebuild  N   ] dev-util/kdbg-1.2.2 to /
[ebuild   U  ] app-text/psutils-1.17 to /
[ebuild   U  ] app-text/a2ps-4.13b-r3 to /
[ebuild   U  ] app-text/jadetex-2.20 to /
[ebuild  N   ] app-text/sgmltools-lite-3.0.3-r2 to /
[ebuild  N   ] kde-base/kdoc-2.2.2-r1 to /
[ebuild  N   ] net-www/htdig-3.1.5-r2 to /
[ebuild  N   ] app-text/enscript-1.6.3-r1 to /
[ebuild  N   ] kde-base/kdebase-2.2.2-r2 to /
[ebuild  N   ] app-doc/qt-docs-2.3.1 to /
[ebuild  N   ] dev-util/kdevelop-2.0.2 to /
```

Mit *N* gekennzeichnete Pakete sind noch nicht auf ihrem Rechner installiert, würden aber durch die angegebene Aktion eingespielt werden. Pakete, die mit einem *U* gekennzeichnet sind, befinden sich bereits in einer älteren Version auf Ihrem System und werden durch diese Aktion aktualisiert.

Folgende Parameter sind ausserdem Verfügbar:

--fetchonly : Lädt alle benötigten Dateien herunter, die für das Kompilieren notwendig sind, sowie alle Abhängigkeiten, die dadurch entstehen.

--emptytree : Diese Option lässt Portage vortäuschen, dass keine der Abhängigkeiten oder Pakete, auf denen das zu installierende Paket beruht, installiert sind. Dies lässt sich sehr gut mit der Option *--pretend* verbinden, um eine komplette Liste der Abhängigkeiten für jedes einzelne Paket anzeigen zu lassen. Alle Abhängigkeiten mit Ausnahme von *glibc* werden dargestellt.

--nodeps : Mit dieser Option versucht Portage nur die angegebene Pakete zu "mergen" und ignoriert sämtliche Abhängigkeiten. Bitte beachten Sie, dass diese Option zu Problemen führen kann, wenn Sie die Pakete, von denen das jeweilige Paket abhängig ist, nicht bereits installiert haben.

--onlydeps : Mit dieser Option ist es möglich, nur die Abhängigkeiten des jeweiligen Paketes zu "mergen", jedoch **nicht** das ausgewählte Paket selbst.

--noreplace : Wenn Sie Pakete zum "mergen" angeben, die bereits installiert sind, Sie jene aber nicht durch neue ersetzen wollen, hilft Ihnen diese Option weiter.

--usepkg : Anstatt das angegebene Paket zu kompilieren, versucht Portage mit dieser Option ein vorkompiliertes tbz2 Paket von einer angegebenen Stelle zu installieren . Jene Stelle ist in der PKGDIR Shellumgebungsvariable anzugeben.

--debug : Um eine noch detailiertere Ausgabe zu bekommen, was während der Aktion mit Portage passiert, benutzen Sie diese Option. Normalerweise werden Ausgaben "menschlich lesbarer" dargestellt. So haben Sie zum Beispiel als Entwickler die Möglichkeit, syntaktische Fehler in den Bash Script basierten ebuild Dateien zu finden.

--autoclean : Zwingt **emerge** zum totalen Bereinigen von paketspezifischen temporären Verzeichnissen für Kompiliervorgänge, bevor das Paket kompiliert wird. Portage erledigt dies bei der standard Konfiguration von selbst, dadurch ist diese Option nur für Entwickler interessant, die dieses Verhalten abgeschaltet haben.

--verbose : Sagt *emerge*, dass es im ausführlichen Modus laufen soll. Zusammen mit *--pretend* werden die möglichen USE Flags ausgegeben.

3.3 "Unmergen" (Deinstallieren) von Paketen

Der Vorgang des "unmergens" ist, dass die Dateien, die mit einem installierten Paket verbunden sind, gelöscht werden. Damit ist die Software vom System entfernt und kann nicht mehr benutzt werden, bis Sie jenes Paket wieder "mergen".

Pakete werden mittels des *emerge* Befehls und dem Parameter *unmerge*, gefolgt vom Namen des Paketes entfernt. Das folgende Beispiel beseitigt alle installierten Versionen vom *ltrace* Paket.

Befehlsauflistung 14: Ein Paket deinstallieren

```
# emerge unmerge ltrace
```

oder

```
# emerge unmerge dev-util/ltrace
```

Portage erlaubt ausserdem das "unmergen" von spezifischen Versionen eines Paketes. Bereiche werden durch = (exakte version), < (kleiner als), > (größer als), <=(kleiner als oder gleich), und >= (größer als oder gleich) dargestellt. Das folgende Beispiel würde alle Versionen, die gleich und älter des Paketes ltrace in der Version 0.3.15 sind, "unmergen".

Befehlsauflistung 15: Bestimmte Versionen eines Pakets deinstallieren

```
# emerge unmerge \<=dev-utils/ltrace-0.3.15
```

Wenn Sie Bereiche für Pakete benutzen, so stellen Sie sicher, das Sie jeweils für die Zeichen > und < ein Backslash davorsetzen, sodass Ihre Shell in diesem Fall dies nicht falsch interpretiert. Ausserdem ist es von Nöten, die Kategorie des Paketes, wie im Beispiel gezeigt, anzugeben. Für andere Beispielausdrücke zu Bereichen, führen Sie den Befehl *emerge --help* aus.

Warnung
Das "unmergen" von Paketen kann gefährlich sein. Wenn Sie ein Paket des Grundsystems entfernen, verliert Ihr System an Funktionalität und bei entfernten Bibliotheken droht funktionsuntüchtige Software. **Portage warnt Sie nicht, wenn Sie Pakete des Grundsystems oder gar Abhängigkeiten anderer Pakete entfernen.**

Wenn das zu entfernende Paket tatsächlich installiert ist, wird das Programm emerge exakt anzeigen, welche Pakete entfernt werden und wartet eine gewisse Anzahl an Sekunden ab, um den Benutzer die Möglichkeit zu geben, den Vorgang mittels der Tastenkombination Control-C abbzubrechen.

Beginnt erstmal der Vorgang des "unmergens", sehen Sie eine lange Liste von Dateinamen, die mit dem Paket verbunden sind. Manche dieser Dateinamen haben ein Merkmal (flag), das an der linken Seite der Datei angezeigt wird. Die Merkmale *!mtime*, *!empty*, und *cfgpro* verdeutlichen, weshalb einige Dateien nicht entfernt worden sind, als das Paket "unmerged" wurde. Dateien ohne jegliche Merkmale wurden erfolgreich vom System entfernt.

Das Merkmal *!mtime* sagt aus, dass die Datei nach der Installation des Pakets geändert wurde. Das bedeutet, dass jemand nach dem "mergen" des Paketes diese Datei bearbeitet hat oder zu einem späteren Zeitpunkt andere Pakete sie überschrieben haben. Dies erlaubt es, dass Pakete aktualisiert werden können, ohne die Gefahr, dass wichtige Dateien entfernt werden.

Das Merkmal *!empty* weist auf Verzeichnisse hin, welche Portage verbietet zu entfernen, da sie nicht leer sind (mehrere Pakete teilen sich oft das selbe Verzeichnis, welches das Paket, was "unmerged" wird, entweder selbst gehört oder ebenfalls benutzt). Der Konfigurationsdatei Schutz-Mechanismus tritt dann ein, wenn Sie das *cfgpro* Merkmal sehen. Das bedeutet, ein neueres Paket, was installiert wird, übernimmt den Besitz jener Konfigurationsdateien und Portage verweigert die Entfernung dieser Dateien.

Warnung
Dateien werden immer dem letzten installierten Paket zugeordnet. Dies ist abhängig von der Reihenfolge der Installation und unabhängig von der aktuellen Versions- oder Revisionsnummer der Pakete, die installiert sind. Wenn ein Paket eine Datei besitzt, wird diese immer mit deinstalliert, auch wenn eine ältere Version eines Paketes diese Datei installiert hat, solange der Nutzer diese Datei nicht manuell geändert hat.

3.4 System Update

Portage unterstützt die Möglichkeit installierte Pakete mit einem einzigen Befehl zu aktualisieren. Das System-Update-Feature ermöglicht es Ihnen, die Kernpakete ihres Systems zu Versionen zu aktualisieren, die von den Gentoo-Entwicklern empfohlen werden und zum einwandfreien Betrieb von Gentoo Linux notwendig sind. Ein System-Update aktualisiert nur Pakete, die als essentiell angesehen werden. Also nur die Pakete, die im Portage Profil angegeben sind, werden als absolut wichtig für den Betrieb und die Aktualität des Systems erachtet.

Um ein System-Update zu starten, geben sie den folgenden Befehl ein:

Befehlsauflistung 16: Das Kernsystem aktualisieren

```
# emerge --update system
```

Portage wird nun, abhängig von den Versionen und Paketen, welche sie derzeit installiert haben, die Updates kompilieren und installieren, die vom aktuellen Portage-Profil empfohlen werden. Sie haben die Möglichkeit sich mit der Option **--pretend** eine Liste mit den Paketen die installiert bzw. aktualisiert werden, ausgeben zu lassen, wenn das oben gezeigte Beispiel ausgeführt würde.

Notiz
Wie sie aus der Gentoo-Installationsanleitung erfahren können, ist einer der ersten Schritte der Befehl *emerge system*, um das Grundsystem zu installieren. Mit *emerge --update system* werden diese Basispakete auf den aktuellsten Stand gebracht.

3.5 World Update

Portage unterstützt außerdem die Möglichkeit, alle nicht-essentiellen Pakete mit einem einzigen Befehl zu aktualisieren. Das Portage-System besitzt dafür einen gewissen Grad an "Intelligenz", die es ermöglicht, ein System mit verschiedene Versionen von Paketen, die in Konflikt zueinander stehen, sicher zu aktualisieren .

Das world-update-Feature von Portage überprüft das Systemprofil, die Liste der blockierten Pakete (package.mask), das world-Profil und die Abhängigkeiten (inkl. Versionkontrolle) von Paketen, die im world-Profil stehen. Dadurch findet es heraus, welche Pakete aktualisiert werden müssen. Ein Paket wird nur aktualisiert, wenn es eine neue Version gibt und das Paket im world-Profil aufgeführt wird oder ein anderes Paket, das im world-Profil steht von ihm abhängt. Selbstverständlich darf das Paket oder eine spezielle Version dessen nicht durch das Systemprofil oder die package.mask blockiert sein.

Portage versucht nun alle Pakete, die im world-Profil aufgeführt sind auf die neuste verfügbare und unblockierte Version zu aktualisieren. Desweiteren überprüft Portage auch die Abhängigkeiten von jedem Paket im world-Profil und wird versuchen diese auf die neuste Version zu aktualisieren. Dabei wird eine Versionkontrolle durchgeführt, sodass die Versionhierachie bestehen bleibt. Außerdem dürfen diese Pakete weder durch das Systemprofil noch durch package.mask blockiert sein. Schließlich werden die *SLOTs* überprüft, die in einem vorangegangenen Kapitel besprochen wurden.

Benutzer, die andere Distributionen und ihre Paket-Management-Systeme neben Portage kennen, sind vielleicht etwas darüber irritiert, dass Portage nicht nur ein blindes Aktualisieren der Pakete vornimmt, einfach nur anhand der Versionsnummern (Wie es bis Gentoo 1.0 gehandhabt wurde).

Viele der Pakete, die in Gentoos Portage-Tree sind, stehen in verschiedenen Versionen zur Verfügung. Eine ältere oder neuere Version eines Paketes kann mit der Software, die auf sie aufbaut inkompatibel sein. Blindes Aktualisieren von Bibliotheken und Programmen, ohne Rücksicht darauf, dass sie von anderen Paketen gebraucht werden, kann schnell zu vielen schwerwiegenden Problemen führen. Um dies zu verhindern lässt Portage beim Aktualisieren Vorsicht walten und bezieht die Abhängigkeiten aller Pakete, basierend auf den Angaben in den einzelnen ebuilds, mit ein.

Das Herz von Portages-World-Update ist das World-Profil. Anders als das System-Profil, welches nur von den Entwicklern gewartet wird und nie vom Benutzer verändert werden sollte, wird das World-Update-Profil indirekt mit der Zeit durch Aktionen des Benutzers erstellt. Das world-Profil funktioniert in etwa wie eine "Favoriten-Liste". Pakete die vom Benutzer manuell mit Hilfe von *emerge* installiert werden, werden in der Datei world aufgezeichnet. Diese Datei findet sich unter /var/cache/edb/world. Portage macht dies, da sie ihm mitgeteilt haben es zu installieren (per emerge) und es annimmt, dass sie ein Interesse daran haben, das Paket immer auf dem aktuellsten Stand zu halten.

Die world Datei besteht aus einem Paketnamen mit Kategorie pro Zeile und sollte in etwa wie folgt aussehen:

Befehlsauflistung 17: Inhalt einer world Datei

```
net-im/gaim
net-www/skipstone
net-www/galeon
app-editors/vim
app-text/ispell
net-mail/evolution
dev-util/ltrace
sys-fs/xfsprogs
=net-www/mozilla-0.9.8-r3
sys-apps/attr
```

```
sys-apps/dmapi
sys-kernel/linux-sources
sys-apps/acl
app-office/gnucash
app-cdr/xcdroast
```

Nahezu alle Einträge in diesem Beispiel wurden von Portage automatisch hinzugefügt, als der Benutzer eines der Pakete manuell "ein-merge-te". Diese Pakete werden aktualisiert, wenn eine neuere Version verfügbar ist.

Notiz
Um Zeit zu sparen und sicher zu stellen, dass alle Ihre bevorzugten Pakete aktuell gehalten werden, können Sie die **world** Datei selbst bearbeiten und so Einträge für diese Pakete hinzufügen. Wenn Sie eine älteren Version von Portage aktualisieren, müssen sie das world-Profil erstellen und dem System bekannt machen. Bei aktuellen Installationen von Gentoo und Portage sollte das world-Profil während der Installation erzeugt werden.

Ein interessanter Eintrag ist der für das Mozilla-Paket (*=net-www/mozilla-0.9.8-r3*). Dieser Eintrag wurde von Hand hinzugefügt, um eine exakte Version festzulegen. Paketeinschränkungen, wie sie im Abschnitt "Unmergen (Deinstallieren) von Paketen" in diesem Handbuch besprochen wurden, können dazu verwendet werden, Portage zu zwingen nur spezielle Versionen beim Aktualisieren von Paketen zu verwenden. Der obige Eintrag hat z.B. den Effekt, dass Portage auf das Paket mozilla-0.9.8-r3 als einzig verfügbare Version festgelegt ist. Somit wird dieses Paket im Verlaufe eines World-Update nie aktualisiert werden.

World-Updates werden durch den folgenden Befehl gestartet:

Befehlsauflistung 18: World Update ausführen

```
# emerge --update world
```

Portage wird nun versuchen alle Pakete die in der world Datei stehen und (wenn nötig) deren Abhängigkeiten aktualisieren. Abhängigkeiten werden auf die neueste verfügbare Version, die vom zu aktualisierenden Paket gebraucht wird, gebracht. Pakete die nicht in der world Datei aufgeführt sind und keine Abhängigkeiten von den voher genannten Paketen sind, werden nicht aktualisiert.

Wenn *emerge* gegen Ihren Willen einige Pakete downgraden will, benutzen Sie *--upgradeonly*:

Befehlsauflistung 19: Erlaube nur das Upgraden

```
# emerge --upgradeonly world
```

Warnung
Portage wird keine Dateien in Verzeichnissen überschreiben, die durch die "Configuration File Protection" (Schutz von Konfigurationsdateien) geschützt sind. Es ist notwendig, dass Sie selbst die Unterschiede zwischen Ihren bestehenden und den neuen Dateien, die von Portage generiert wurden, ausgleichen. Wenn Sie Ihre Konfigurationsdateien nicht aktualisieren, werden verschiedene Programme nicht mehr funktionieren. Bitte schauen sie für weitere Informationen unter "Schutz der Konfigurationsdateien" im Kapitel "Portage konfigurieren" nach oder benutzen sie den Befehl *emerge --help config*.

Um eine Liste mit den Paketen zu sehen, die von einem World-Update betroffen sind, können Sie das Argument **--pretend** verwenden, so wie es bereits in einem vorangegangen Abschnitt in diesem Kapitel besprochen wurde.

Notiz
Durch ein World-Update wird gleichzeitig auch ein System-Update durchgeführt. Außerdem können Kernpakete nicht auf bestimmte Versionen in der world Datei festgelegt werden, da sie vom aktuellen Portage-Profil immer überschrieben werden!

Ein praktischer Nebeneffekt der Art wie World-Update arbeitet, ist für Benutzer interessant, die ein komplettes neu-Kompilieren aller installierten Pakete auf einem System wünschen. Da World-Update alle Pakete und deren Abhängigkeiten, die in der Datei world stehen aktualisiert, gibt einem die Option *--emptytree* die Möglichkeit eine neu-Kompilierung sämtlicher Pakete und aller Abhängigkeiten - mit Ausnahme der glibc - zu erzwingen. Das ist z.B. für Leute nützlich, die ihre Compiler-Optionen oder ihre USE Variable geändert haben und wollen, dass diese Veränderungen von der gesamten Software die sie benutzen verwendet wird - ohne dass sie nun jedes Paket selbst erneut "mergen" müssen. Dazu müssen sie einfach die Datei **world** mit allen Paketen, die sie verwenden auffüllen und den folgenden Befehl verwenden:

Befehlsauflistung 20: Das System komplett neu kompilieren

```
# emerge --update world --emptytree
```

Sie können die Option *--pretend* mit diesem Befehl verwenden, um eine Liste mit den Paketen, welche neu-Kompiliert werden, zu bekommen.

3.6 System aufräumen

Portage hat die Fähigkeit verschiedene Versionen eines Paketes parallel zu installieren. Es gibt einige Pakete in Gentoos Portage-Tree die diese Funktion nutzen (z.B. zur Kompatibilitätssicherung, wenn ältere Programme mit neueren Versionen inkompatibel sind).

Denken Sie daran, dass wenn eine neuere Version eines Paketes installiert wird, in den meisten Fällen ein Großteil des älteren Paketes überschrieben wird und alles was zurückbleibt sind einige Dokumentationsdateien und andere für das System unwichtige. Mit der Zeit können diese "Dateileichen" sehr viel Festplattenplatz verschwenden.

Um dies zu verhindern bietet Portage einen einfachen Weg an, Rückstände veralteter Dateiversionen vom Benutzersystem zu entfernen. Diese Funktionalität ergibt sich aus der emerge-Option *clean* und kann folgendermaßen benutzt werden:

Befehlsauflistung 21: Das System aufräumen

```
# emerge clean
```

emerge wird nun eine Liste mit Paketversionen und -revisionen ausgeben die entfernt werden und die Versionen die erhalten bleiben. Außerdem gibt es dem Benutzer Zeit, die Aktion mit Control+C abzubrechen. Auf einem normalen System werden nun eine Vielzahl von Aktionen durchgeführt, die eine lange Listen mit Dateien, die entweder gelöscht oder erhalten wurden, ausgibt.

Naheliegenderweise wird Portage die Aufräumaktion auf die world Datei (alle installierten Pakete) anwenden. Sie können den Umfang der Säuberung durch Optionen wie *world*, *system*, eine Liste von Paketnamen oder eine Einschränkung auf Paketversionen, wie es im Abschnitt "Unmerge" in diesem Kapitel besprochen wurde, beeinflussen.

Beim Herausfinden, welche Paketversionen entfernt werden sollen, überprüft Portage die verschiedenen Profile, die Beziehungen zu anderen Paketen und den SLOT eines Paketes. Vorausgesetzt dass alle Paketabhängigkeiten für alle Pakete richtig definiert sind, wird *emerge clean* nur veraltet Pakete vom System entfernen und nicht solche, deren Entfernung die Funktionalität des Systems beeinträchtigen würde.

3.7 Pakete säubern

Portage bietet außerdem die Funktion ein Paket zu säubern (engl. *prune*). *prune* ist eine unsichere Variante von *clean*. Es entfernt alle Versionen aller Pakete, ausgenommen der zuletzt installierten Version. Es führt nur wenige Überprüfung aus, die *clean* durchführt und kann grundlegende Abhängigkeiten von Ihrem System entfernen! Wenn Sie diese Option nutzen, können sie sehr schnell ihr System unbrauchbar machen. Somit wird diese Variante nicht empfohlen und sollte nur in wenigen Ausnahmenfällen verwendet werden.

Die Aktion *prune* akzeptiert die selben Optionen wie die Aktion *clean* und kann wie folgt angewendet werden:

Befehlsauflistung 22: Pakete säubern

```
# emerge prune
```

3.8 Den Portage-Tree durchsuchen

Portage-Trees, wie jener der das Herzstück von Gentoo Linux bildet, können sehr groß sein. Der Befehl **emerge** bietet eine Suchfunktion an, die Suchanfragen in Form eines regulären Ausdrucks, dieser muss von Anführungszeichen eingeschlossen sein, akzeptiert. Reguläre Ausdrücke sind sehr komplizierte Biester und Anwendern die sich dafür interessieren, sei ein gutes Buch zum Thema empfohlen.

Die meisten einfachen Suchen könne ohne Wissen, wie ein regulärer Ausdruck zu bilden ist durchgeführt werden. Das folgende ist ein Beispiel für eine einfache Suche nach einem Paket, das "gcc" heisst oder "gcc" im Namen hat:

Befehlsauflistung 23: Nach ebuilds suchen

```
# emerge search gcc
```

Für jeden Treffer gibt der Befehl den Paketnamen, die neuste Version, die neuste installierte Version, seine Homepage und eine Beschreibung über die Software im Paket aus.

3.9 Hilfe erhalten

Mehr Informationen über die zahlreichen Optionen und Aktionen die *emerge* unterstützt, erhalten Sie durch die Eingabe von:

Befehlsauflistung 24: Hilfe zu emerge aufrufen

```
# emerge --help
```

3.10 Nützliche Werkzeuge

Verschiedenste Werkzeuge wurden von Gentoo-Benutzern erstellt, um das Arbeiten mit *emerge* zu erleichtern. Diese Werkzeuge sind im Paket *app-admin/gentoolkit* im Gentoo-Portage-Tree zu finden.

- *etc-update* : Shell-Skripte für vim, die dabei helfen die Dateien in /etc abzugleichen (eine falsche Benutzung kann gefährlich sein!)
- *qpkg* : Werkzeug für Datenbankanfragen
- *epm* : ein weiteres Anfragewerkzeug für die Datenbank mit RPM-ähnlicher Syntax
- *etcat* : zeigt und erläutert die USE Flags für ein Paket

4. Was sind maskierte Pakete (masked packages)?

Viele Leute sind verwundert, warum ein neu erschienenes Paket nicht bei einem *emerge -u world* enthalten ist. Eine gutes Beispiel ist xfree-4.3.0 (die Version zur Zeit des Schreibens). Wenn Sie *emerge sync* gefolgt von einem *emerge -u world* ausführen, werden Sie kein Update zu xfree in der Liste sehen. Warum?

Der Grund ist, dass bestimmte Pakete als "masked" markiert sind. Das Paket wird nicht automatisch auf eine neuere Version gebracht oder installiert, solange Sie nicht selbst Hand anlegen. Für eine Beschreibung, wie Sie die Installation von maskierten Paketen durchführen, möchten Wir Sie auf die Masked Packages FAQ in unserem Gentoo Forum verweisen.