


Paketmanagement und –distribution bei freien Unices



Copyright © by Benjamin Schweizer <gopher at h07 dot org>
<http://www.redsheep.de/>

1. Inhaltsverzeichnis

1.	Inhaltsverzeichnis	2
2.	Abbildungsverzeichnis	3
3.	Paketmanagement und -distribution bei freien Unices	4
4.	Eine kurze Einführung... ..	6
5.	Was ist ein Paket?	7
5.1.	Das RPM Format	7
5.2.	Das DEB Format.....	8
5.3.	Aufbau von Tarballs	9
6.	Quellpakete und Binärpakete	11
6.1.	Binärpakete.....	11
6.2.	Quellpakete.....	12
7.	Paketmanager und Paketmanagement	13
7.1.	RPM und seine Helfer.....	13
7.2.	dpkg/APT	16
7.3.	Paketmanagement bei den BSD-Unices	19
7.4.	Portage System	22
8.	Resümee	24
9.	Anhang	25
9.1.	Glossar	25
9.2.	Quellenverzeichnis	26

2. **Abbildungsverzeichnis**

Tabelle 1: Metainformationen bei RPM Paketen	8
Tabelle 2: Metainformationen bei Debian Paketen.....	9
Tabelle 3: Metainformationen in Tarballs	10
Tabelle 4: Paketbeschreibung eines RPM Pakets	15
Tabelle 5: Paketbeschreibung eines Debian Pakets	18

3. Paketmanagement und -distribution bei freien Unices

Mehr als 30 Jahre nach seiner Erstveröffentlichung durch die Bell Labs hat das portable Betriebssystem Unix eine beachtliche Verbreitung erreicht¹. War es zu Beginn ein Clone des Multics-Betriebssystems und für den Einsatz auf großer und teurer Hardware konzipiert, hat sich das in den 1990er Jahren geändert. Inzwischen arbeiten Unices und Unix-derivate wie z.B. Linux² auf einem breiten Spektrum an Hardware: von Embedded Geräten mit MIPS oder StrongARM CPU bis hin zu den ehemaligen Mainframes der IBM zSeries oder HP Superdomes.

In seiner bewegten Geschichte hat das „Ur-Unix“ mehrfach seinen Besitzer gewechselt und wurde an viele Unternehmen und Universitäten lizenziert. Diese trugen maßgebliche Teile zu dem bei, was die verschiedenen Unix-Derivate heute repräsentieren. Einen wichtigen Zweig stellt die an der University of Berkeley gepflegte Berkeley Software Distribution, kurz BSD, dar. Aus der Version 4.3BSD ging im Sommer 1990 4.3BSD Lite hervor, eine Version in der sämtlicher geschützter Quelltext ersetzt wurde. Aus diesem ersten freien Release wurden verschiedene neue Distributionen abgeleitet, von denen NetBSD³, FreeBSD⁴ und OpenBSD⁵ die bekanntesten Vertreter darstellen.

In dieser Zeit gewann auch das von der Free Software Foundation⁶ (FSF) vorangetriebene Betriebssystem GNU an Popularität. Richard Stallmans Ziel war es mit GNU, was ein Akronym auf „Gnu's Not Unix“ ist, einen freien Clone von Unix unter der General Public License⁷ (GPL) zu etablieren. Mit Linux⁸, dem ebenfalls unter der GPL veröffentlichten Kernel des Finnen Linus Torvalds, wurde die letzte fehlende Komponente hinzugefügt. Das

¹ Vgl. [Tan01], Seite 13

² Vgl. [Kof98], Seite 31 ff.

³ Vgl. [NetBSD04]

⁴ Vgl. [FreeBSD04]

⁵ Vgl. [OpenBSD04]

⁶ Vgl. [FSF04]

⁷ Vgl. [GPL91]

⁸ Vgl. [Kof98]

GNU/Linux getaufte Betriebssystem wird seit dem über verschiedene Distributionen verbreitet, wobei die ältesten und bekanntesten RedHat Linux⁹, SuSE Linux¹⁰ und Debian GNU/Linux¹¹ darstellen.

Diese breite Verfügbarkeit von freien und kommerziellen Unices hat zu einer Verlagerung der Einsatzgebiete geführt. Die Entwicklung zu immer kleineren und flexibleren Systemen kehrt das Bild vom Unix-Administrator um. Waren in den 1970ern oft mehrere hochqualifizierte Administratoren für die Betreuung eines einzelnen Servers zuständig, entfallen heute gleich mehrere Systeme je Administrator. Um mit den gestiegenen Anforderungen Schritt zu halten, bedient man sich heute mächtiger Werkzeuge um die Kontrolle über das System zu behalten. Eines dieser Werkzeuge ist das Paketmanagement. Welche Implementierungen es bei den genannten Unices und Linux Distributionen gibt und worin die jeweiligen Stärken und Schwächen liegen, werde ich in dieser Arbeit erörtern.

⁹ Vgl. [RHL04]

¹⁰ Vgl. [Sus04]

¹¹ Vgl. [Deb04]

4. Eine kurze Einführung...

Moderne Unix-Systeme sind heute oft in vernetzten Umgebungen beheimatet. Sicherheitslücken sind viel bedrohlicher als bei Standalone Systemen, auf denen die Anwender dem Administrator bekannt sind. Es ist daher notwendig Software regelmäßig zu aktualisieren oder neu aufzuspielen. Dabei soll das produktive System so wenig wie möglich beeinträchtigt werden. Traditionell erfolgt solch ein Update durch eine erneute Source-Installation, also dem Übersetzen des Quelltextes. Dieses Vorgehen lässt sich dabei auf folgende Punkte reduzieren:

- Entpacken des Archivs (Tarball)
- Konfigurieren der Verzeichnisse und Bibliotheken
- Übersetzen des Quellcodes
- Installieren der Binaries
- Entfernen temporärer Dateien

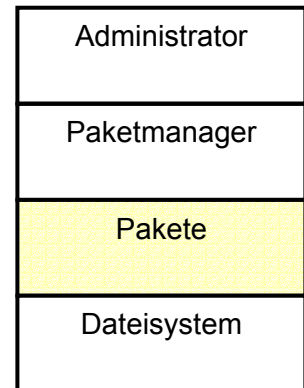
Eine Source-Installation ist durchaus akzeptabel, geht es um die Installation von wenigen Programmpaketen. Stehen jedoch tägliche Updates ins Haus, benötigt man eine automatisierbare Lösung. Hier setzt die Idee des Paketmanagements an.

Allgemein beschreibt der Begriff Paketmanagement alle notwendigen Schritte um Software, Dokumentation oder andere Systemdateien zu installieren, deinstallieren, aktualisieren oder reparieren. Zentrale Komponente dabei ist der jeweilige Paketmanager, er hat die Aufgabe den Zugriff auf die einzelnen Pakete zu abstrahieren und dadurch eine einheitliche Schnittstelle zur Verfügung zu stellen.

Zu den wesentlichen Aufgaben eines Paketmanagers gehören neben dem einfachen Entpacken und Erstellen eines Paketes auch Versionsverwaltung, die Behandlung von Abhängigkeiten zwischen Paketen, die Kontrolle von Prüfsummen oder die Überprüfung von Signaturen. Dabei variiert der Funktionsumfang je nach Strategie und Entwicklungsgrad der jeweiligen Implementierung.

5. Was ist ein Paket?

Pakete (im Englischen mit „packages“ und nicht „packets“ übersetzt) gibt es schon sehr lange. Grundsätzlich handelt es sich dabei um Archive, wie sie auch mit den Kommandos „ar“, „compress“ oder „tar“ erzeugt werden. Ihre Aufgabe ist es alle zu einem Programm gehörenden Dateien zu verpacken und so eine einfache Verteilung oder Sicherung zu ermöglichen. Abhängig vom Paketformat kann über Systemtools auf den Inhalt eines Paketes zugegriffen werden und es wird nicht zwingend der jeweilige Paketmanager benötigt.



5.1. Das RPM Format

Der Red Hat Package Manager (RPM) wurde 1995 von Red Hat entwickelt. Er verwaltet RPM-Pakete (.rpm), wie sie bei Red Hat-, SuSE- und Mandrake Linux¹² eingesetzt werden. Diese Pakete enthalten Programmdateien (Binaries), die direkt nach der Installation verwendet werden können. Ein Übersetzen (kompilieren) ist nicht notwendig. Der Programmquelltext ist optional in so genannten Source-RPMs verfügbar (.src.rpm) und wird nur von wenigen Anwendern verwendet. Speziell die GPL zwingt die Autoren eines Paketes, die Packager, jedoch dazu, neben einer Binär-Version auch den Quelltext anzubieten.

Pakete im RPM-Format enthalten neben den eigentlichen Dateien zusätzliche Metainformationen, die vom Paketmanager während der Installation bzw. Deinstallation verwendet werden:

Bezeichnung	Erklärung
Name	Programmname
Version	Programmversion
Release	Paketversion

¹² Vgl. [Man04]

Summary u. Description	Beschreibung
Group	Kategorie, z.B. Netzwerk
Size	Gesamtgröße der enthaltenen Dateien
Packager	Autor des Pakets
Build Date	Erstelldatum
License	Lizenzbestimmungen (häufig GPL oder BSD)
Prüfsumme	MD5-Prüfsumme der enthaltenen Dateien
Signatur	PGP Signatur des Pakets
Dependencies	Für den Betrieb notwendige Pakete
Conflicts	Pakete die sich gegenseitig ausschließen
Provides	Virtuelle Pakete
Install- und Uninstall-Skripte	Skripte, die bei der Installation/Deinstallation des Paketes ausgeführt werden
Trigger Skripte	Skripte, die bei der Installation/Deinstallation eines anderen Paketes aufgeführt werden
Verify Skripte	Skripte zur Überprüfung der Installation
Ghosts	Virtuelle Dateien, die erst nach der Installation erzeugt werden
Documentation Flag	Besondere Markierung der Dokumentation

Tabelle 1: Metainformationen bei RPM Paketen

5.2. Das DEB Format

Die komplett freie Linux-Distribution Debian GNU/Linux setzt auf ihr eigenes Paketformat, das DEB-Format. Es wurde im Jahr 1993 vom Debian-Gründer Ian Murdock entwickelt. Im Gegensatz zum RPM-Format wird bei Debian-Paketen nicht zwischen Quell- und Binär-Paketen unterschieden; dies erfolgt wie auch das Lizenzmanagement über die zugrunde liegende Verzeichnisstruktur. Alle Pakete im selben Verzeichnisbaum unterliegen vergleichbaren Lizenzbestimmungen. Neben dem Standard-Verzeichnis „main“ gibt es die Verzeichnisse „non-free“ für Software unter nicht-freien Lizenzen, „contrib“ für Software, die von nicht-freier Software abhängig ist, und „non-us“ für Programme, die Exportbeschränkungen aus den USA unterliegen (für gewöhnlich Software, die starke Kryptographie implementiert).

Bei den Paketen selbst handelt es sich um Archive des Archivierers „ar“, die einem bestimmten Aufbau entsprechen. Sie enthalten neben den Programmdateien diese Metainformationen:

Bezeichnung	Erklärung
Package	Paketname
Priority	Priorität des Paketes (möglich: required, standard, optional)
Section	Kategorie, z.B. Netzwerk
Installed-Size	Größe nach Installation
Maintainer	Autor des Pakets
Architecture	Hardware-Architektur (z.B. i386, Alpha)
Version	Version des Pakets
Provides	Virtuelle Pakete
Depends	Für den Betrieb notwendige Pakete
Recommends, Suggests	Empfohlene Pakete
Conflicts	Pakete die sich gegenseitig Ausschließen
Size	Größe des Pakets
MD5Sum	Prüfsumme des Pakets
Description	Beschreibung des Pakets
Install- und Uninstall-Skripte	Skripte, die bei der Installation/Deinstallation des Paketes ausgeführt werden

Tabelle 2: Metainformationen bei Debian Paketen

5.3. Aufbau von Tarballs

Das dritte bedeutende Paketformat bei freien Unices und Derivaten stellen Tarballs (.tar, .tgz, .tar.gz, .tar.bz2) dar. Tarballs sind Archive des weit verbreiteten Archivierers „tar“, die zumeist mit einem zusätzlichen Packer komprimiert werden. Ihr interner Aufbau kann im Gegensatz zu Debian- und RPM-Paketen stark variieren. Sie werden für gewöhnlich direkt vom Entwickler erstellt und sind nicht für einen speziellen Paketmanager optimiert. Sie

werden im Kontext des Paketmanagements meist zur Verteilung von Quellcode eingesetzt, der auf dem Zielsystem ohnehin noch übersetzt werden muss.

Bei Tarballs haben sich über die Jahre gewisse Standards etabliert, die mit den Metainformationen von Debian- und RPM Paketen vergleichbar sind. Sie sind nicht zwingend notwendig, werden aber von vielen Entwicklern als sinnvoll erachtet:

Datei	Erklärung
AUTHORS	Liste der Autoren/Entwickler
COPYING oder LICENSE	Lizenzbestimmungen
ChangeLog oder HISTORY	Unterschiede zu vorangegangenen Versionen
INSTALL	Installationsanweisungen
README	Wichtige Informationen und Programmbeschreibung
TODO oder BUGS	Bekannte Fehler
VERSION	Programmversion
SRC/*	Quelltext
MAN/*	Dokumentation (Manuals)
DOC/*	Zusätzliche Dokumentation

Tabelle 3: Metainformationen in Tarballs

6. Quellpakete und Binärpakete

Der traditionelle Weg der Software-Installation unter Unix besteht aus dem Übersetzen des Programm-Quelltextes auf dem jeweiligen Zielsystem. Dies ist ein zeitintensiver Vorgang und erfordert das Vorhandensein eines Compilers nebst aller verwendeten Bibliotheken. Gerade auf kleinen und dadurch meist schwachen Rechnern ist dies kein optimaler Zustand. Es scheint daher interessant diese Aufgabe auszulagern.

6.1. Binärpakete

Einen Ansatz, um das Übersetzen vom Produktionsrechner loszulösen, stellen Binär-Installationen dar. Die Programme werden dabei auf einem separaten Entwicklungssystem übersetzt und anschließend in Binärform auf das Zielsystem übertragen. Diesen Weg verfolgen unter anderem die Distributionen Red Hat Linux, SuSE Linux, Mandrake Linux und Debian GNU/Linux. Die Software wird bei ihnen direkt vom jeweiligen Distributor bzw. dem Autoren eines Paketes für die entsprechende Architektur übersetzt und als Binärpaket weitergegeben.

Binärpakete sind für den Anwender oft einfacher in der Handhabung als Quellpakete, bedeuten aber für den Packager einen erheblichen Aufwand. Dieser muss für jede Hardware-Architektur eigene Pakete bereitstellen - bei Debian GNU/Linux sind das aktuell zwölf¹³. Innerhalb einer Architektur besteht weiterhin die Möglichkeit spezielle Prozessor-Features wie 32/64 Bit, MMX, ISSE oder SMP zu verwenden und für jede Unterversion auch ein eigenes Paket anzubieten. Selbst auf der gleichen Hardware können weitere Optionen im Programm aktiviert oder deaktiviert werden, beispielsweise kann der Apache Webserver optional mit oder ohne SSL-Unterstützung übersetzt werden. Dies bezeichnet man auch als unterschiedliche „Flavours“.

Es ist für die einzelnen Distributoren auf Grund der vielfältigen Unterscheidungsmöglichkeiten kaum möglich alle Kombinationen abzudecken. Sie legen sich stattdessen auf gewisse Mindestanforderungen fest und bieten nur die wesentlichen Programme in verschiedenen Versionen an. Diese Mindestanforderungen werden eher konservativ gewählt, um

¹³ Vgl. [Deb04], Verzeichnis /ports/

abwärtskompatibel zu älterer Hardware zu bleiben und verzichten auf spezielle Prozessor-Optimierungen.

Neben Binärpaketen bieten diese Distributionen auch zu beinahe jedem Paket ein Source-Paket an. Dies liegt an den restriktiven Bestimmungen der GNU General Public License und ähnlicher Open Source Lizenzen¹⁴. Sie zwingen den Autoren eines Paketes zur Weitergabe des Quelltextes, auch wenn dieser oft nicht benötigt wird. Er wird meist in einem zusätzlichen Quellpaket bereitgestellt und kann bei Bedarf eingesehen und selbst übersetzt werden.

6.2. Quellpakete

Einen anderen Ansatz verfolgen die Distributionen Gentoo Linux¹⁵, FreeBSD, NetBSD und OpenBSD. Bei ihnen sind Sourceinstallationen ein fester Bestandteil des Betriebssystems. Der Administrator kann dazu an zentraler Stelle festlegen, welche Prozessor-Features er verwenden möchte und neu übersetzte Programmpakete sind optimal an die zugrunde liegende Hardware angepasst.

Für die Distributoren ist dies ein großer Gewinn. Sie stellen lediglich eine „Bauanleitung“ bereit, die genau beschreibt welche Patches neben dem eigentlichen Quellcode eingespielt werden müssen, wie der Quelltext und diese Patches bezogen werden können und wie die zugrunde liegende Verzeichnisstruktur aussieht. Die eigentliche Übersetzungsarbeit erfolgt auf dem Rechner des Anwenders. Die Distributoren sparen dadurch Speicherplatz, Bandbreite und Rechenzeit und beschränken die Software nicht in ihrer Funktion, in dem sie spezielle Flavours festlegen.

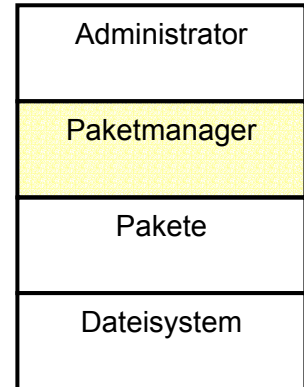
In jüngerer Zeit lässt sich bei Gentoo Linux und den freien BSD Unices der Trend zu vermischten Installationen beobachten. Für viele Standard-Pakete bieten sie zusätzlich zu den „Bauanleitungen“ auch Binär-Pakete an. In Abhängigkeit von der eingesetzten Architektur variiert diese Zahl jedoch erheblich, und besonders auf exotischer Hardware führt oft kein Weg vorbei um eine Installation aus den Quellen.

¹⁴ Vgl. [OSI04]

¹⁵ Vgl. [Gen04]

7. Paketmanager und Paketmanagement

Nach dem in den vorangegangenen Kapiteln die verschiedenen Paketformate und der Unterschied zwischen Quellpaketen und Binärpaketen behandelt wurde, beschäftigt sich dieses Kapitel mit dem eigentlichen Paketmanager. Der Paketmanager hat die Aufgabe den Umgang mit Paketen zu abstrahieren. Er stellt eine Schnittstelle zum Administrator beziehungsweise zu weiteren Programmen dar und kapselt die wesentlichen Funktionen: Installation, Deinstallation, Aktualisierungen, Suchanfragen in der Paketdatenbank und Überprüfung von installierten Paketen.



7.1. RPM und seine Helfer

Bei den auf dem RPM-Paketformat basierenden Distributionen wird der gleichnamige Paketmanager RPM¹⁶ (Red Hat Package Manager) eingesetzt. Er ist unter den Lizenzbestimmungen der GPL frei verfügbar und wird neben Red Hat Linux auch bei SuSE- und Mandrake Linux eingesetzt. Seit seiner ersten Veröffentlichung 1995 wurden einige verbesserte Versionen veröffentlicht, aktuell ist die Version 4.0.4. Eine vollständige Beschreibung der Syntax findet sich im Anhang. An dieser Stelle möchte ich nur auf die wesentlichen Funktionen und besonderen Merkmale eingehen:

- Die Installation von Programmpaketen erfolgt durch das Kommando `rpm -ivh foobar-1.0-1.i386.rpm`. Dabei werden Abhängigkeiten (Dependencies) zu anderen Paketen ermittelt, die eventuell vorher installiert werden müssen. Neben diesen Abhängigkeiten werden auch gegenseitige Ausschlüsse (Conflicts) berücksichtigt. Es macht beispielsweise keinen Sinn mehrere Mailserver gleichzeitig zu betreiben und diese schließen sich gegenseitig aus. Während der Installation wird zuerst ein Pre-Install Skript aufgeführt und anschließend ein Post-Install Skript.

¹⁶ Vgl. [RPM04]

Dadurch können sehr genaue Eingriffe in Konfigurationsdateien anderer Programme vorgenommen werden.

- Das Deinstallieren von Paketen erfolgt mit dem Kommando `rpm -e foobar`. RPM überprüft, ob dieses Paket für den Betrieb anderer Pakete notwendig ist. Falls keine Abhängigkeiten bestehen, wird das Paket samt aller Konfigurationsdateien entfernt. Wie bei der Installation gibt es auch hier Pre- und Post-Uninstall Skripte, die eine saubere Deinstallation sicherstellen.
- Grosses Augenmerk legten die RPM Entwickler auf eine Update-Funktion: `rpm -Uvh foobar-1.0-2.i386.rpm`. Es werden die Pre-/Post-Install Skripte ausgeführt und Konfigurationsdateien können so oft übernommen werden. Eine Besonderheit bei RPM sind auch so genannte Trigger-Skripte. Sie werden ausgeführt, wenn ein anderes Paket installiert/deinstalliert wird. So kann beispielsweise ein Trigger-Skript des Apache Webserver die Konfiguration verändern, wenn die Skriptsprache PHP installiert wird.
- RPM ermöglicht es alle installierten Dateien mit den Original-Versionen aus den Installations-Paketen zu vergleichen. Das Kommando `rpm -Vp foobar-1.0-1.i386.rpm` überprüft die installierten Dateien und hilft eventuell gelöschte oder veränderte Dateien ausfindig zu machen.
- RPM pflegt eine Datenbank, in der alle installierten Pakete und Dateien verzeichnet sind. über diese Datenbank können sehr elegant Suchanfragen ausgeführt werden. `rpm -q foobar` gibt die Version des installierten Paketes „foobar“ zurück, `rpm -qf /etc/foobar.conf` verrät zu welchem Paket die Datei „/etc/foobar.conf“ gehört. Ein besonderes Feature von RPM-Paketen sind so genannte Ghosts-Dateien. Sie sind in den Paketen nicht vorhanden, werden aber direkt bei der Installation erzeugt. Durch ihre besondere Markierung können auch sie einem Paket zugeordnet werden.

Neben dem eigentlichen Paketmanager kommen bei den einzelnen Distributionen einige weitere Hilfsprogramme zum Einsatz. Ihre Aufgabe ist es Systemupdates und Sicherheitsupdates effizient und automatisiert durchzuführen. Bei Red Hat Linux wird dazu das Programm `up2date` verwendet. Es stellt eine https-Verbindung zu den Red Hat Update Servern (Red Hat Network¹⁷) her und überprüft ob neue Versionen von lokal installierten Paketen verfügbar sind. Diese können dann entweder „von Hand“ über `up2date --install` installiert oder über eine Weboberfläche für einen späteren Zeitpunkt eingeplant werden.

YaST Online Update (YOU) ist das Update Programm bei SuSE Linux. Es präsentiert sich als Komponente von YaST¹⁸, einer grafische Administrationsoberfläche. YOU vergleicht die lokale Paketdatenbank mit online verfügbaren Versionen und verwendet RPM um neue Versionen einzuspielen.

Mandrake Linux verwendet für Systemupdates das Programm `urpmi`¹⁹. `urpmi` ist eine Programmsuite die die Funktionen von RPM kapselt und dem Anwender eine einfache Schnittstelle präsentiert. Das Kommando `urpmi.update` gleicht dabei die installierten Programmversionen mit einem RPM-Repository ab und spielt neuere Versionen ein.

7.2. *dpkg/APT*

Der Debian Paketmanager²⁰ (`dpkg`) gilt als einer der mächtigsten Paketmanager. Er wurde 1993 von Ian Murdock, dem Gründer des Debian-Projekts entwickelt und wird neben Debian GNU/Linux auch beim Fink-Projekt²¹ für Apple Mac OS X eingesetzt. Die meisten von RPM bekannten Funktionen finden sich auch hier:

- Pakete können sehr einfach installiert werden. Ist das Programmpaket auf einem eingebundenen Datenträger, üblicherweise der Festplatte oder einem CD-ROM,

¹⁷ Vgl. [RHN04]

¹⁸ Vgl. [Sus01], Seite 104 ff.

¹⁹ Vgl. [Man04], Verzeichnis `/cooker/urpmi.html`

²⁰ Vgl. [Pen03], Kapitel 14.1

²¹ Vgl. [Fink03]

geschieht das mit dem Kommando `dpkg -i foobar.deb`. Dabei werden Abhängigkeiten (Dependencies) zu anderen Paketen ermittelt und der Administrator darauf hingewiesen. Eine Besonderheit bei `dpkg` ist die Unterscheidung zwischen Empfehlungen (Recommends) und Erweiterungen (Suggests).

- Die Deinstallation erfolgt über das Kommando `dpkg -r foobar`. Wie auch bei RPM wird das Paket nur dann deinstalliert, wenn keine anderen Pakete es benötigen. Bei der Deinstallation bleiben die Benutzereinstellungen in `*/etc/*` erhalten. Sollen wirklich alle Dateien, einschließlich der Konfigurationsdateien gelöscht werden, erfolgt dies durch den zusätzlichen Parameter `--purge`, kurz `-P`: `dpkg -rP foobar`.
- Das Aktualisieren bereits installierter Pakete erfolgt wie eine Installation durch das Kommando `dpkg -i foobar.deb`. Im Unterschied zu einer Neuinstallation werden die alten Programmdateien gesichert und können im Falle eines Fehlers wiederhergestellt werden. Für den Fall, dass Änderungen an den Konfigurationsdateien stattgefunden haben und der Paket- Verwalter ebenfalls eine neue Version bereitgestellt hat, wird der Administrator darauf hingewiesen und kann entscheiden, ob er die eigenen Dateien behalten will oder ob er die Version vom Packager installieren möchte.
- Wie bei RPM können mit dem Debian Paketmanager die Installierten Dateien mit ihren Originalen aus den Paketen abgeglichen werden und eventuell beschädigte/gelöschte Dateien werden angezeigt. Dies erfolgt mit dem Befehl `debsums -a`. Sollten tatsächlich Dateien fehlen genügt, eine Neuinstallation des beschädigten Paketes.
- Die `dpkg` Datenbank unterstützt sehr mächtige Suchanfragen. Reguläre Ausdrücke²² erlauben eine Suche mit Wildcards. Das Kommando `dpkg -S foobar\.conf` listet alle Pakete auf, die die Datei „foobar.conf“ beinhalten. Eine Suche nach der Datei „foobar-123.conf“ könnte durch `dpkg -S foo.*\.conf` erfol-

²² Vgl. [Kof98], Seite 363 ff.

gen. Es werden dabei alle Dateien aufgelistet, die zwischen „foobar“ und „.conf“ beliebige Zeichen beinhalten.

- Eine detaillierte Paketbeschreibung zu installierten Paketen erhält man mit dem Kommando `dpkg --info foobar`. Den Inhalt nicht installierter Pakete kann man mit dem Kommando `dpkg --contents foobar.deb` anzeigen.

```
new debian package, version 2.0.
size 195058 bytes: control archive= 750 bytes.
    667 bytes,   17 lines   control
    194 bytes,    8 lines   * postinst      #!/bin/sh
    131 bytes,    6 lines   * preinst       #!/bin/sh
Package: rsync
Version: 2.5.5-0.1
Section: net
Priority: optional
Architecture: i386
Depends: libc6 (>= 2.2.4-4), libpopt0 (>= 1.6.2-1)
Suggests: ssh
Installed-Size: 364
Maintainer: Philip Hands <phil@hands.com>
Description: fast remote file copy program (like rcp)
 rsync is a program that allows files to be copied to and from remote
 machines in much the same way as rcp. It has many more options than
 rcp, and uses the rsync remote-update protocol to greatly speed up
 file transfers when the destination file already exists.
.
The rsync remote-update protocol allows rsync to transfer just the
differences between two sets of files across the network link.
~
~
(END)
```

Tabelle 5: Paketbeschreibung eines Debian Pakets

Neben dem Paketmanager spielt die Programmbibliothek APT eine zentrale Rolle in der Debian Update Strategie. Die Bibliotheks- Funktionen von APT werden über das Programm `apt-get`²³ bereitgestellt. `apt-get update` kann von verschiedenen Quellen, zu- meist HTTP- und FTP-Servern, aber auch lokalen Datenträgern, eine Liste aller verfügbaren Pakete laden und diese mit der lokalen Paketdatenbank vergleichen. Anschließend werden diese neuen Versionen mit dem Befehl `apt-get upgrade` eingespielt.

²³ Vgl. [Pen03], Kapitel 14.3

7.3. *Paketmanagement bei den BSD-Unices*

Die freien Unices FreeBSD, NetBSD und OpenBSD verwenden für Programminstallationen häufig die direkt von den Entwicklern bereitgestellten Tarballs, wie sie in Kapitel 5.3 beschrieben sind. Sie sind in der Regel mit dem Ziel der Plattformunabhängigkeit erstellt worden und enthalten den Quelltext einer Applikation. In den Original-Quellen sind daher auch keine Zusatzinformationen wie Abhängigkeiten oder Hinweise auf Konflikte mit ihnen hinterlegt, diese Metainformationen finden sich in den „Bauanleitungen“ wieder. Diese „Bauanleitungen“ nennt man bei FreeBSD und OpenBSD „ports“ oder auch Ports-Collection²⁴ und bei NetBSD „packages“ bzw. Packages-Collection²⁵.

Die Ports/Packages finden sich in der lokalen Verzeichnishierarchie unter „/usr/ports/“ bzw. „/usr/pkgsrc/“ wieder. Dort sind für jedes Programmpaket „Makefiles“ hinterlegt, die vom Programm „make“ ähnlich wie Skripte abgearbeitet werden. Makefiles enthalten alle wesentlichen Informationen um ein Programm zu übersetzen: URL des Quelltextes und zusätzlicher Patches, Compiler-Flags, Abhängigkeiten zu weiteren Paketen und genaue Verzeichnisangaben. Im Umgang unterscheiden sich Ports und Packages nicht wesentlich von Binärpaketen, die grundlegenden Funktionen finden sich auch hier wieder:

- Um ein Programm zu installieren genügt es in das entsprechende Verzeichnis zu wechseln, beispielsweise „/usr/pkgsrc/editors/joe“ und dort das Kommando `make` auszuführen. Make berücksichtigt eventuelle Abhängigkeiten und installiert benötigte Bibliotheken vor dem eigentlichen Programm. Anschließend lädt es den Programmquelltext und zusätzliche Patches. Stimmen die Prüfsummen der einzelnen Tarballs, werden Quelltext und Patches ausgepackt und übersetzt. Durch das Kommando `make install` werden die Programmdateien schließlich installiert. Dabei kopiert FreeBSD die Programmdateien direkt an die entsprechenden Stellen im Verzeichnisbaum, NetBSD und OpenBSD erzeugen zuerst ein Binär-Paket und installieren im Anschluss dieses. Die temporären Dateien die während der Übersetzung erzeugt wurden, können mit dem Befehl `make clean` wieder entfernt werden.

²⁴ Vgl. [FreeBSD_ports] und [OpenBSD_ports]

²⁵ Vgl. [NetBSD_packages]

- Die Deinstallation erfolgt analog zur Installation über das Kommando `make deinstall` im jeweiligen Verzeichnis der Ports-/Packages-Collection. Das Makefile enthält dazu alle notwendigen Anweisungen und entfernt alle Dateien die zu einem Port/Package gehören.
- Sollen Pakete aktualisiert werden, müssen sie dazu erneut übersetzt werden. Es ist erforderlich diese zuerst mit `make deinstall` zu löschen und anschließend die neue Version mit `make` und `make install` zu installieren. Paketupdates entsprechen einer Deinstallation der alten Software und der Installation der neuen.
- Informationen über die Pakete und eine Paketbeschreibung findet man in der Datei `/usr/pkgsrc/README.html` (NetBSD) beziehungsweise in `/usr/ports/index.html` (FreeBSD, OpenBSD) oder direkt online bei den Distributoren.
- Spezielle Optimierungen, die Steuerung von optionalen Funktionen und Richtlinien für unterschiedlich restriktive Lizenzen, die Flavours, können zentral über die Dateien `/etc/make.conf` (FreeBSD) oder `/etc/mk.conf` (NetBSD, OpenBSD) eingestellt werden. Je nach Einstellung wird beispielsweise die OpenSSL Bibliothek verwendet oder es wird auf Verschlüsselung verzichtet.

Von Zeit zu Zeit ist es ratsam die Ports-/Packages-Collection zu aktualisieren. Spätestens wenn Sicherheitsupdates eingespielt werden müssen, ist dies der Fall. Es gibt dabei mehrere Möglichkeiten; die einfachste besteht aus dem Löschen der bisherigen Ports-/Packages-Collection und dem Herunterladen und Entpacken einer neuen Version. Dadurch sind die neuen „Bauanleitungen“ verfügbar und aktualisierte Software kann installiert werden. Dieses Vorgehen ist jedoch recht ineffizient, genügt es doch nur die aktualisierten Komponenten zu installieren. Dies ist über CVS²⁶ möglich: bei einem „Checkout“ werden nur die veränderten Dateien übertragen. Es gibt verschiedene Programme, die diese Methode vereinfachen. Neben dem eigentlichen `cv`s `-co` Kommando wird häufig auch CVSup²⁷ eingesetzt.

²⁶ Vgl. [Col02]

²⁷ Vgl. [Pol02]

Neben einer Quelltext basierten Installation besteht neuerdings auch die Möglichkeit Binärpakete zu installieren. Binärpakete stehen für viele populäre Programme auf verbreiteten Plattformen zur Verfügung, können aber auch leicht selbst gebaut werden. Die wichtigsten Programme im Umgang mit Binärpaketen sind `pkg_add`, `pkg_delete` und `pkg_info`:

- Mit dem Kommando `pkg_add foobar-1.0.0.tgz` wird das Binärpaket „foobar“ in Version 1.0.0 installiert. Dabei überprüft `pkg_add` Abhängigkeiten zu weiteren Paketen und stellt sicher, dass auch diese installiert sind. Eine Besonderheit bei der NetBSD Implementierung von `pkg_add` ist es, dass Pakete direkt von FTP- und HTTP-Servern installiert werden können; sie müssen nicht lokal gespeichert sein.
- Das Deinstallieren von Paketen erfolgt entsprechend über den Befehl `pkg_delete foobar-1.0.0`. Von `pkg_delete` werden dabei Abhängigkeiten berücksichtigt; sollen Pakete mit entfernt werden, die von anderen Programmen benötigt werden, erfolgt dies durch `pkg_delete -f foobar-1.0.0`.
- Das Aktualisieren von Paketen erfolgt wie bei Quellpaketen durch Deinstallieren des alten Paketes und die anschließende Installation des neuen. Dabei ist zu beachten, dass häufig die exakte Versionsnummer eines Paketes mit in den Abhängigkeiten angegeben ist und `foobar-1.0.0` und `foobar-1.0.1` gänzlich unterschiedliche Pakete sind. Einen Ansatz um dieses Problem zu lösen stellt `portupgrade`²⁸ bei FreeBSD dar. Es pflegt eine eigene Paketdatenbank und erzeugt für alte Versionen virtuelle Pakete.
- Auskunft über die installierten Pakete erhält man mit `pkg_info`. Das Kommando listet alle installierten Pakete, ihre Versionsnummern und eine kurze Beschreibung auf.

²⁸ Vgl. [Luc01]

7.4. Portage System

Portage²⁹ heißt das Paketmanagement bei Gentoo Linux. Es ist wie das Paketmanagement bei den BSD Unices auf Quelltext basierte Installationen ausgelegt, unterstützt in neuen Versionen aber auch Binärpakete. Da es sich bei Portage um einen recht jungen Paketmanager handelt, finden sich viele Konzepte von anderen Paketmanagern hier wieder. Die „Bauanleitungen“ für Quellpakete nennen sich bei Gentoo Linux „ebuilds“. Das Programm Emerge ist die zentrale Komponente des Portage System. Es ist Paketmanager und Update-Mechanismus in einem. Seine Funktionen beschreiben sich wie folgt:

- Die Installation neuer Programmpakete erfolgt mit dem Kommando **emerge foobar**. Emerge überprüft mögliche Abhängigkeiten und installiert diese zuerst. Dann werden der Programmquelltext und notwendige Patches geladen, Prüfsumme und Signatur kontrolliert und anschließend wird das Programm lokal übersetzt. Wenn alle Schritte erfolgreich, erzeugt Emerge ein Binärpaket und installiert dieses in den Verzeichnisbaum. Falls es für das Programm „foobar“ ein Binärpaket gibt, kann auch dieses mit dem Befehl **emerge -K foobar** installiert werden.
- Das Deinstallieren von Software erfolgt mit dem Kommando **emerge clean foobar**. Es löscht ein Paket (hier „foobar“), falls es nicht zwingend von anderen Paketen benötigt wird oder zum Basissystem gehört. Soll ein Paket in jedem Falle gelöscht werden, auch wenn dies die Stabilität des übrigen Systems beeinträchtigen kann, muss dies mit dem Befehl **emerge unmerge foobar** erfolgen.
- Das Aktualisieren der Ports-Collection erfolgt mit der Anweisung **emerge sync**. Dabei wird die lokale Verzeichnishierarchie mit dem offiziellen Gentoo Portage Repository abgeglichen. Dieser Abgleich erfolgt über das moderne rsync-Protokoll, bei dem nur Veränderungen übertragen werden, wodurch diese Methode besonders bei langsamen Verbindungen effektiv ist.

²⁹ Vgl. [Gen04], Datei /doc/en/portage-user.xml

- Nach dem die Ports-Collection aktualisiert wurde, können mit dem Kommando `emerge -u foobar` neue Programmversionen eingespielt werden. Dabei entfernt Emerge die alte Programmversion automatisch und nimmt die Installation des neuen ebuids vor. Falls Änderungen an den Konfigurationsdateien vorgenommen wurden, können diese entweder „von Hand“ in die neuen Versionen eingetragen werden oder mit dem optionalen `etc-update` Skript interaktiv übernommen werden.
- Suchanfragen erfolgen im Portage System mit dem Programm `qpkg` aus dem `gentoolkit`³⁰. Eine kurze Paketbeschreibung erhält man mit dem Kommando `qpkg -i foobar`; alle installierten Dateien des Paketes „foobar“ werden mit `qpkg -l foobar` angezeigt. Um das Paket ausfindig zu machen, zu dem die Datei `/etc/foobar.conf` gehört, wird der Befehl `qpkg -f /etc/foobar.conf` verwendet und die Integrität installierter Dateien wird mit `qpkg -cv foobar` geprüft.

³⁰ Vgl. [Gen04], Datei `/doc/en/gentoolkit.xml`

8. Resümee

Die Frage „Welcher Paketmanager ist für mich am Besten geeignet?“ lässt sich nicht ohne weiteres beantworten. Alle behandelten Paketmanager haben ihre individuellen Stärken und Schwächen. Ihre Aufgabe, Programme effizient zu installieren und vollständig zu entfernen, erfüllen sie alle. Die Wahl nach dem richtigen Paketmanager hängt daher wesentlich vom Einsatzgebiet und dem zugrunde liegenden Betriebssystem ab.

Der NetBSD Paketmanager PkgSrc ist auf eine Vielzahl anderer Betriebssysteme portiert worden und bietet damit die Möglichkeit mit dem selben Programm auf unterschiedlichen Betriebssystemen zu arbeiten. Neben freien und kommerziellen Unices unterstützt PkgSrc auch die Microsoft Windows Plattform – dort kann er einen viel versprechenden Ansatz für ein professionelles Paketmanagement bieten. Es macht hingegen wenig Sinn ein bereits vorhandenes Paketmanagement abzulösen, denn die Zahl der verfügbaren Pakete und ihre Qualität lässt sich durch einen „fremden“ Paketmanager nur schwer erzielen.

Das dpkg/APT Gespann von Debian GNU/Linux zählt zu meinen persönlichen Favoriten. Es stellt sehr mächtige Funktionen im Umgang mit Binärpaketen bereit und regelmäßige Updates verlaufen meist völlig problemlos. Leider wird von der Möglichkeit Pakete zu signieren noch wenig Gebrauch gemacht – ein Feature das gerade bei Online Updates unabdingbar sein sollte.

Neben all den technischen Faktoren ist die Wahl des „richtigen“ Paketmanagers auch eine Frage des Geschmacks. Der „falsche“ Paketmanager führt dazu, dass man an der Distribution vorbei administriert und ihr Potential nicht komplett ausschöpft.

9. Anhang

9.1. Glossar

Binär Installation	Eine Programminstallation die dazu fertig übersetzte Binärprogramme verwendet
BSD License	Eine Open Source Lizenz
Compiler Flags	Parameter an den Compiler um spezielle Optimierungen zu steuern
Conflicts	Ausschluss eines anderen Pakets
Dependencies	Abhängigkeiten zu anderen Paketen
Distribution	Alle Komponenten eines Betriebssystems: Kernel, Systemprogramme und Anwendungen
Distributor	Der Hersteller einer Distribution
Flavours	Die Möglichkeit verschiedene Programmfunktionen zu aktivieren / deaktivieren
GPL	GNU General Public License - Eine Open Source Lizenz, die die Weitergabe des Quelltextes vorschreibt
Maintainer	Siehe Packager
make	Programm zu Abarbeitung von Makefiles
Makefile	Eine Art Skript die für Quelltextinstallationen verwendet wird
Package	Ein Programmpaket, häufig enthält es die Programmdateien in Binärform
Package-System	siehe Ports
Packager	Der Verfasser/Autor eines Pakets
Patch	Eine Veränderung am Programmquelltext
Ports	Eine Sammlung von „Bauanleitungen“ für Quelltext-Pakete
Provides	Ein virtuelles Paket wird bereitgestellt
Recommends	Ein weiteres Paket wird empfohlen
Source Installation	Eine Programminstallation die dazu den Programmquelltext verwendet
Suggests	Ein weiteres Paket wird empfohlen
Trigger Skript	Ein Skript das aufgeführt wird, wenn sich der Zustand eines anderen Pakets ändert

Unix	Sammelbegriff für verschiedene Distributionen die auf dem Ur-Unix von AT&T basieren
Unices	siehe Unix

9.2. Quellenverzeichnis

Bücher

- [Kof98] Kofler, Michael: *Linux – Installation, Konfiguration, Anwendung*, 3. Auflage. Addison-Wesley, 1998 – (ISBN 3-8273-1304-X)
- [Mui96] Mui, Linda: *Wenn der Unix-System-Administrator nicht zu finden ist*. O'Reilly/International Thomson Verlag GmbH & Co KG, 1996 – (ISBN 3-930673-23-1)
- [Red99] *Red Hat Linux 7 Referenzhandbuch*. Red Hat, Inc., 1999 – (keine ISBN)
- [Sus01] *SuSE Linux – das Handbuch*, 19. Auflage. SuSE GmbH, 2001 – (ISBN 3-934678-25-4)
- [Tan01] Tanenbaum, Andrew S.: *Modern Operating Systems*, 2. Auflage. Prentice-Hall, Inc., 2001 – (ISBN 0-13-031358-0)
- [Tor01] Torvalds, Linus: *Just for Fun*. HarperCollins Publisher, Inc., 2001 – (ISBN 3-446-21684-7)

Online Quellen

- [Col02] CollabNet, Inc.: *Concurrent Versions System*. Abfrage v. 7. Feb. 04 – (<http://www.cvshome.org/>)
- [Deb04] *Debian GNU/Linux*, Abfrage v. 7. Feb 04 – (<http://www.debian.org/>)
- [FreeBSD04] *The FreeBSD Project*, Abfrage v. 7. Feb. 04 – (<http://www.freebsd.org/>)
- [FreeBSD_Ports] The FreeBSD Project: *FreeBSD Ports*. Abfrage v. 7. Feb. 04 – (<http://www.freebsd.org/ports/>)
- [Fin03] *The Fink Project*. Abfrage v. 7. Feb. 04 – (<http://fink.sourceforge.net/>)
- [FSF04] *Free Software Foundation, Inc.*, Abfrage v. 7. Feb. 04 – (<http://www.fsf.org/>)
- [Gen04] Gentoo Technologies: *Gentoo Linux*. Abfrage v. 7. Feb. 04 – (<http://www.gentoo.org/>)
- [GPL91] Free Software Foundation: *GNU General Public License*, 1991 – (<http://www.gnu.org/copyleft/gpl.html>)

- [Luc01] Lucas, Michael: *Cleaning Up Ports*. Version v. 29. Nov. 03 –
(http://www.onlamp.com/pub/a/bsd/2001/11/29/Big_Scary_Daemons.html)
- [Man04] Mandrake Soft: *Mandrake Linux*. Abfrage v. 7. Feb. 04 – (<http://www.linux-mandrake.com/>)
- [NetBSD04] *The NetBSD Project*, Abfrage v. 7. Feb. 04 - (<http://www.netbsd.org/>)
- [NetBSD_Packages] The NetBSD Project: *The NetBSD Packages Collection*. Abfrage v. 7. Feb. 04 – (<ftp://ftp.netbsd.org/pub/NetBSD/packages/pkgsrc/README.html>)
- [OpenBSD04] *The OpenBSD Project*, Abfrage v. 7. Feb. 04 – (<http://www.openbsd.org/>)
- [OpenBSD_Ports] The OpenBSD Project: *Ports*. Abfrage v. 7. Feb. 04 –
(<http://www.openbsd.org/ports.html>)
- [OSI04] Open Source Initiative: *Licensing*. Abfrage v. 7. Feb. 04 –
(<http://www.opensource.org/licenses/>)
- [Pen03] Pennington, Havoc: *Debian Tutorial*. Version 14. Feb. 03 –
(<http://www.debian.org/doc/manuals/debian-tutorial/>)
- [Pol02] Polstra, John D.: *CVSup Home Page*. Abfrage v. 7. Feb. 04 –
(<http://www.cvsup.org/>)
- [RHL04] *Red Hat Inc.*, Abfrage v. 7. Feb. 04 – (<http://www.redhat.com/>)
- [RHN04] Red Hat, Inc.: *About Red Hat Network*. Abfrage v. 7. Feb. 04 –
(<https://rhn.redhat.com/>)
- [RPM04] *RPM Package Manager*, Abfrage v. 7. Feb. 04 – (<http://www.rpm.org/>)
- [Sus04] *SuSE Linux AG.*, Abfrage v. 7. Feb. 04 – (<http://www.suse.com/>)