Debian-Paketmanagement

Ed. 1

VERSIONSGESCHICHTE

NUMMER	DATUM	BESCHREIBUNG	NAME
	2016-06-30T17:29:32+00:00		

Inhaltsverzeichnis

Zusammenfassung

Die Debian-Distribution setzt sich aus mehreren zehntausend Bausteinen zusammen, die alle aufeinander abgestimmt sind und sich bei Bedarf in eine Installation integrieren. Diese sogenannten Pakete (Packages) sind so eigenständig, dass sie von einem oder mehreren Debian-Entwicklern für das Debian-Projekt gepflegt werden, interagieren aber zugleich so intensiv mit allen anderen, dass wechselseitige Abhängigkeiten erkannt und bei Bedarf automatisch aufgelöst werden. Nur so ist die Modularität des komplexen Gesamtsystems gewährleistet, die Administratoren weltweit die Möglichkeit bietet, Debian-Installationen sehr genau für die jeweilige Anforderung vom Embedded-Gerät über den Desktop bis zum Großrechner zu konfigurieren.

Effizientes Paketmanagement ist also für jeden Debian-Administrator ein ebenso interessantes wie lohnendes Feld, das in der Praxis aber oft nicht ausreichend beachtet und mit wenigen Standardbefehlen "erledigt" wird. Zwei ausgewiesene Debian-Experten nehmen dies zum Anlass, das Debian-Paketmanagement erstmals derart umfassend darzustellen. Das Buch kommt von den Konzepten, die der Struktur und dem Zusammenspiel der Pakete zugrunde liegen, über die Werkzeuge zu deren Nutzung immer auch zu den Best Practices der professionellen Systemadministration. Es wendet sich an Einsteiger ebenso wie an Berufsadministratoren, indem es, ausgehend von den Grundlagen, das Optimierungspotential in zunehmend umfangreichen Szenarien ausschöpft. So entsteht ein aktuelles Handbuch der Debian-Administration, das als praxisorientiertes HowTo ebenso dient wie als Nachschlagewerk für die unerwartet zahlreichen Optionen und Kombinationsmöglichkeiten.

Über dieses Buch

Kann Paketmanagement Spaß machen?

Ja! Und wir werden Ihnen in diesem Buch zeigen, warum das so ist.

Software ist heute meist sehr komplex und darum modular aufgebaut. Das gilt nicht nur für das Betriebssystem Linux und andere freie Anwendungen, sondern hat sich als allgemeines Prinzip in der Softwareentwicklung durchgesetzt.

Modularität hat mehrere Facetten: einzelne Bausteine für spezifische Aufgaben, klare Beschreibungen zu deren Funktion, definierte Schnittstellen und Protokolle zur Kommunikation untereinander. All dies gewährleistet die Kombination und Austauschbarkeit von Komponenten, also die flexible Anpassung der Software an konkrete Anforderungen. Modularität heißt aber auch Abhängigkeiten: Bausteine und Funktionen bedingen einander, bauen aufeinander auf, verlangen bei der Installation eine vorgegebene Reihenfolge – und stehen ggf. zueinander in Konflikt. Das betrifft insbesondere Varianten und Entwicklungsstufen einer Implementierung.

Auf die Verwaltung von Software übertragen, heißt das: Die einzelnen Module werden als *Pakete* (*Packages*) bereitgestellt. Das setzt voraus, dass deren Bezug zueinander (*Relation*) klar geregelt ist; nur so kann ein Betriebssystem wie Debian GNU/Linux (siehe Abschnitt 1.1) funktionieren und weiterentwickelt werden, an dem Hunderte Entwickler aus der ganzen Welt mitwirken und das inzwischen aus mehr als 40.000 Paketen besteht. Ohne ein leistungsfähiges Paketmanagement wäre dies unmöglich.

Debian GNU/Linux und davon abgeleitete Betriebssysteme – wie Ubuntu [Ubuntu], Linux Mint [LinuxMint], Knoppix [Knoppix] oder Grml [Grml] – setzen auf dem Paketformat `deb` und der Paketverwaltung mit `dpkg` und APT auf. Neben dem RPM-Paketformat (siehe Abschnitt 2.2) ist die Kombination aus dem `deb`-Format und seinen Werkzeugen am weitesten unter den verschiedenen Linux-Distributionen verbreitet. Das hat mehrere Gründe:

- Es funktioniert verlässlich.
- Es ist ausführlich und meist auch verständlich dokumentiert. Leider ist die Dokumentation aber nicht ganz einheitlich und recht verstreut – weshalb nicht zuletzt auch dieses Buch entstanden ist.
- Pakete für Debian GNU/Linux sind aufeinander abgestimmt, wurden vorab intensiv getestet und unterliegen strengen Qualitätskontrollen.
- Pakete für Debian GNU/Linux werden nach ihrer Veröffentlichung (*Release*) bzw. ihrem Entwicklungszweig kategorisiert: *oldoldstable*, *oldstable*, *stable*, *testing*, *unstable* oder *experimental*. Ein Paket für Debian GNU/Linux kann in mehreren dieser Zweige parallel vorliegen und unterscheidet sich nur in seinem jeweiligen „Reifegrad“. Als Benutzer wissen Sie daher genau, worauf Sie sich einlassen, wenn Sie einen bestimmten Entwicklungsstand benutzen (falls nicht, lesen Sie in Abschnitt 2.10 nach). Das Debian-Derivat namens Ubuntu handhabt das etwas anders: Es unterscheidet nur zwischen mehreren stabilen Veröffentlichungen und dem Entwicklungszweig. Im Rahmen einer halbjährlichen Freigabe wird aus dem Entwicklungszweig die nachfolgende, stabile Veröffentlichung.
- Kein Stress mit Lizenzen. Es ist klar geregelt, welche Bedingungen ein Paket erfüllen muss, damit es überhaupt in den offiziellen Bestand von Debian GNU/Linux unter den Distributionsbereich *main* Eingang findet. Alle anderen Pakete werden in die Bereiche *contrib* oder *non-free* einsortiert. Ubuntu kennt kein Äquivalent zu *contrib* und verwendet statt *non-free* die beiden Bereiche *restricted* und *multiverse* (siehe Abschnitt 2.9).
- Die beiden Debian-Entwicklungszweige *unstable* und *testing* (siehe Abschnitt 2.10) wie auch der Bereich *Debian Backports* (siehe Kapitel 19) bekommen regelmäßig neue Pakete, die das Paketverwaltungswerkzeug `aptitude` (siehe Abschnitt 6.3.2) in einer eigenen Liste übersichtlich darstellt. Das ist fast wie Weihnachten, nur günstiger und häufiger.

All dies gewährleistet zwar nicht, dass Software fehlerfrei ist, allerdings reduziert dieses Vorgehen die Zahl der Fehlerquellen deutlich. Es stellt insbesondere sicher, dass sich Softwarepakete unter Berücksichtigung ihrer Abhängigkeiten konfliktfrei installieren, konfigurieren, ausprobieren und auch wieder vollständig aus dem System entfernen lassen. Der Fall, dass andere, bereits integrierte Komponenten Schaden nehmen, ist bei korrektem Vorgehen nahezu ausgeschlossen. Falls das Problem doch auftritt, ist es definitiv in überschaubarer Zeit mit Bordmitteln zu beheben. Diese Werkzeuge stehen im Mittelpunkt dieses Buches.

Die Sorge, dass Sie durch Ausprobieren Ihr Arbeitsgerät unbenutzbar machen, ist unberechtigt – zumindest innerhalb von Debian *stable*. Aber auch in Debian *unstable* passiert das nur sehr selten. Ausführlicher gehen wir darauf im Zusammenhang mit Distributionsbereichen (siehe Abschnitt 2.9) und Veröffentlichungen (siehe Abschnitt 2.10) ein. Fühlen Sie sich also ausdrücklich ermutigt, mit den Paketen Ihres Debian-Systems zu experimentieren!

Zum Buch

Über die Autoren

Dipl.-Inf. Axel Beckert [Beckert-Webseite] hat Informatik mit Nebenfach Biologie an der Universität des Saarlandes studiert. Er arbeitet als Linux-Systemadministrator an der ETH Zürich, ist im Vorstand der Linux User Group Switzerland (LUGS) und Mitglied des Debian-Projekts. Er benutzt `aptitude` seit über 10 Jahren und ist seit der Neuformierung des `aptitude`-Teams zum Jahreswechsel 2011/2012 als Mentor, Versuchskaninchen für neue Versionen und Paketsponsor bei `aptitude` mit an Bord. Seit 2015 ist er auch offiziell einer der Maintainer des Pakets `aptitude` und kümmert sich primär um die Paketierung.

Dipl.-Inf. Frank Hofmann hat Informatik mit Nebenfach Englisch an der Technischen Universität Chemnitz studiert. Derzeit arbeitet er in Berlin im Büro 2.0 [Buero2.0], einem Open-Source Experten-Netzwerk, als Dienstleister mit Spezialisierung auf Druck und Satz [Hofmann-Webseite]. Weiterhin ist er Mitgründer des Schulungsunternehmens Wizards of FOSS [Wizards-of-Foss]. Seit 2008 koordiniert er das Regionaltreffen der Linux User Groups aus der Region Berlin-Brandenburg und gehört zu den Ansprechpartnern rund um die Community-Plattform `lug.berlin` [lug.berlin]. Seit Sommer 2013 ist er Mitglied im Debtags-Projekt [Debian-Debtags], das sich die Verschlagwortung der einzelnen Debian-Pakete zum Ziel gesetzt hat.

Wie und warum dieses Buch entstand

Das Thema „Paketmanagement“ beschäftigt uns als Autoren schon sehr lange. Obwohl jeder die Werkzeuge und Mechanismen tagtäglich verwendet, entdeckten wir zunächst unabhängig voneinander immer wieder neue Aspekte, die sich schrittweise zu einem komplexen Gesamtbild ergänzten.

Beim gemeinsamen Fachsimpeln entstanden aus dieser Begeisterung heraus zunächst Beiträge für die Zeitschrift LinuxUser [Hofmann-Osterried-Alien-LinuxUser] [Hofmann-Winde-Aptsh-LinuxUser] [Hofmann-Debtags-LinuxUser]. Parallel dazu arbeiteten wir weitere Aspekte digital auf und veröffentlichten entsprechende Blogbeiträge [Beckert-Blog], hielten Vorträge bei Linux-Veranstaltungen und versuchten uns in einem Screencast zum Thema.

Im Herbst 2012 hatte Axel die Idee, einen LinuxUser-Artikel zu `aptitude` im Alltagsgebrauch zu schreiben. Dazu kam es bisher noch nicht[1], denn eine Reihe von Vorarbeiten war dazu notwendig. Wir einigten uns daher auf einen Beitrag zu den Unterschieden zwischen `apt-get` und `aptitude`, der jedoch immer länger und länger wurde und schließlich im Frühjahr 2013 in einen Zweiteiler mündete [Beckert-Hofmann-Aptitude-1-LinuxUser] [Beckert-Hofmann-Aptitude-2-LinuxUser].

Bevor wir uns daran machten, Passagen aus diesen umfangreichen Beiträgen wieder herauszustreichen, fiel irgendwann der Satz: „Wenn wir so weitermachen, können wir eigentlich gleich ein Buch schreiben“. Seitdem ließ uns diese Idee nicht mehr los. Teile der Texte und Abbildungen wurden aus den erwähnten Veröffentlichungen übernommen und nach Bedarf für das vorliegende Werk überarbeitet. Das Ergebnis halten Sie nun in Ihren Händen.

Motivation

Uns fasziniert die Paketverwaltung unter Debian, deren Mächtigkeit und unglaubliche Robustheit. Sie funktioniert so klaglos, dass man schon wieder skeptisch werden müsste und nach konzeptionellen Fehlern sucht – aber es gibt tatsächlich keine. Wie in jedem größeren IT-Projekt gibt es neben den intensiv genutzten, gut dokumentierten Bereichen aber auch „dunkle Ecken“

[1] Jörg, bitte nicht böse sein!

und unangenehme Bugs, kuriose Lösungen und kurzfristige Workarounds; es sind allerdings nur wenige, die auch nur in recht ausgefallenen Situationen zutage treten.

Genießen Sie also das beruhigende Gefühl, dass bei der Verwendung der Werkzeuge eigentlich nichts schiefgehen kann – und wenn doch, gibt es immer einen kurzen Weg, das Malheur wieder zu beseitigen.

Sich hingegen in dem vielschichtigen Geflecht aus `dpkg`, APT und `aptitude` zurechtzufinden und ein Verständnis für die einzelnen Programme und Mechanismen zu entwickeln, bedarf Ihrerseits ein wenig Geduld: Ohne nachzulesen und intensiv auszuprobieren, geht es nicht – und auf eben diesem Weg möchte Sie unser Buch begleiten.

Nach einem ersten, flüchtigen Blick auf die genannten Werkzeuge zur Paketverwaltung scheint es so, als sei es unerheblich, welches wann zum Einsatz kommt. Dem ist nicht so, denn jedes hat seine ureigene Aufgabe in der Hierarchie der Paketverwaltung. Subtile Unterschiede zwischen APT und `aptitude` sorgen mitunter für eine blutige Nase, und insbesondere Ein- und Umsteiger aus der RPM-Welt haben es zu Beginn nicht so leicht. Daher gibt es im Anhang eine Übersicht zu den Aufrufen von RPM und YUM — siehe Kapitel 46.

Das Buch will darum vor allem Klarheit schaffen und Ihnen die Zusammenhänge zwischen den Programmen deutlich machen. Es hilft Ihnen, in jeder Situation das passende Werkzeug zur Paketverwaltung auszuwählen und es gekonnt einzusetzen. Dazu fassen wir den aktuellen Stand der Entwicklung zusammen und beleuchten darüber hinaus die angrenzenden Programme bzw. die damit verbundenen Fragestellungen im Alltag der Systembetreuung.

Technische Basis

Rein technisch setzt das Buch auf AsciiDoc [AsciiDoc] auf — einem Textformat, aus welchem dann über mehrere Zwischenstufen diverse Ausgabeformate wie PDF, EPUB oder HTML entstehen. Basierend auf einer einzigen Quelle stehen damit passende Ergebnisse für die verschiedenen Ausgabegeräte zur Verfügung. Die AsciiDoc-Dateien liegen in einem Versionskontrollsystem namens Git und sind auf der Plattform GitHub verfügbar [dpmb-github]. Neben der Möglichkeit, während des Arbeitens auch auf eine frühere Revision zurückgreifen zu können, ermöglicht das ein paralleles, verteiltes Arbeiten von verschiedenen Standorten aus. Zudem kann jeder Interessierte am Buch in Form von Vorschlägen und Korrekturen beitragen.

Versionsverwaltung mit Git
Den Einstieg zu Git erleichtert Ihnen das gleichnamige Buch von Julius Plenz und Valentin Haenel (Julius Plenz und Valentin Haenel: Git. Verteilte Versionsverwaltung für Code und Dokumente, Open Source Press, München, 2. Auflage November 2014, ISBN 978-3-95539-119-5).

Quellcode und Lizenz

Der o.g. Quellcode des Buches findet sich auf GitHub [dpmb-github] und ist unter der Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 4.0 International Lizenz [CreativeCommons] frei verfügbar.

Änderungswünsche oder -vorschläge zum Buch senden Sie bitte dort als Issue [github-issue] — oder sogar noch besser — als Pull-Request mitsamt Patch [github-pull-request] ein.

Organisatorisch

Beide Autoren leben und arbeiten etwa 90 Flugminuten (oder rund 850 km) voneinander entfernt – in Zürich und Berlin. Regelmäßige Arbeitstreffen waren daher nur begrenzt möglich und wurden mit Hilfe von Buchsprints sowie elektronischer Kommunikation überbrückt. Das Buch entsteht seit dem Frühjahr 2013 und häufig auch im Rahmen von Linux-Events. Besonders hervorzuheben sind hierbei die Chemnitzer Linux-Tage [CLT], die Rencontres Mondiales du Logiciel Libre [RMLL] und die Debian Entwicklerkonferenz [DebConf]. An diesen Veranstaltungen nehmen wir gern aktiv teil und nutzen die Gelegenheit, das Buch gemeinsam zu vervollständigen.

Viele Texte verfassen wir zudem von unterwegs aus. Die bisherigen Stationen umfassen Andorra, Augsburg, Beauvais (Picardie), Bergneustadt, Berlin, Besançon, Bottighofen (Bodensee, Schweiz), Bruchsal, Chemnitz, Biel, Delémont, Essen, Frankfurt/Main, Freiburg im Breisgau, Friedrichshafen, Genf, Germersheim, Goizueta (Baskenland, Spanien), Hamburg, Heidelberg, Koblenz,

Lauchringen (Baden, Wutachtal), Lausanne, Meersburg, Montpellier, München, Port del Cantó (Katalanische Pyrenäen, Spanien), Radebeul bei Dresden, Rostock-Warnemünde, Saint Cergue (Jura, Schweiz), Saint-Jouin-Bruneval (Normandie), Tübingen, Zernez (Engadin) und Zürich (siehe Abbildung 1). Wir nahmen uns dabei an der Philosophie von Debian GNU/Linux ein Beispiel: ohne Hektik, mit dem Blick fürs Detail und zumeist pedantisch bis ins letzte i-Tüpfelchen, aber trotzdem mit viel Freude, Neugierde und unserem Entdeckerdrang folgend.

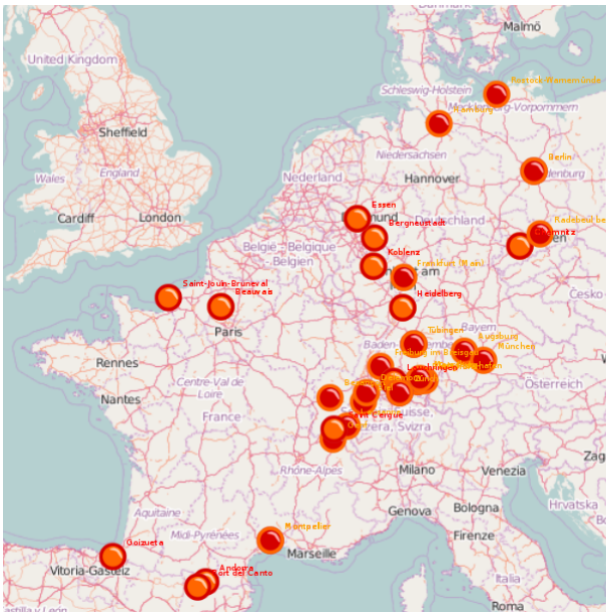

Abbildung 1: Orte, an denen das vorliegende Buch entstand

Grundlagenwissen für Administratoren

Der sichere Umgang mit der Paketverwaltung zählt zu Ihrem Grundwissen als Administrator, um ein UNIX-/Linux-System einrichten und in Bezug auf die eingesetzte Software betreuen zu können. Betreiben Sie Ihre Systeme als Benutzer in Eigenverantwortung, sind diese Kenntnisse für Sie im Alltag ebenso unverzichtbar.

Unabdingbar ist die Auseinandersetzung mit dem Paketmanagement für Zertifizierungen. Das Linux Professional Institute (LPIC) widmet dem Thema in der Zertifikatsstufe LPIC-1 einen eigenen Schwerpunkt mit hoher Gewichtung [lpic-101].

Material für Ihre LPIC-Prüfungen
Ihre Vorbereitung auf die anspruchsvollen Tests des LPI ergänzen die beiden Bücher „LPIC-1. Vorbereitung auf die Prüfungen des Linux Professional Institute“ von Peer Heinlein [Heinlein-LPIC-1] und „LPIC-1. Sicher zur erfolgreichen Linux-Zertifizierung“ von Harald Maaßen [Maassen-LPIC-1].

Dokumentation zu `aptitude`

Das vorliegende Buch resultiert auch aus einem Ärgernis, das zur weltweit verteilten Zusammenarbeit über das Netz gehört: Das Internet vergisst nichts, und irgendwo ist immer noch eine veraltete Dokumentation verlinkt, deren Hinfälligkeit mangels Verfallsdatums auch nicht zu erkennen ist.

Bei der Recherche nach `aptitude`-Optionen verzweigen Suchtreffer häufig auf unklare, überholte und vielfach verteilte Erläuterungen. Als erster Anhaltspunkt bei einer überschaubaren Fragestellung mag das helfen, kann aber auch in eine Sackgasse oder gar zu Fehlern führen, wenn sich die Software just in diesem Punkt weiterentwickelt hat.

Der Wunsch nach einem aktuellen, konsistenten und einsprachigen Nachschlagewerk zur Paketverwaltung mit `dpkg`, APT und `aptitude` erhielt also ausreichend Nahrung, zumal auch die an recht prominenter Stelle verlinkte Online-Dokumentation zu

aptitude veraltet war (Stand: 2008). Auf Axels Initiative wurde sie aber mittlerweile auf den neuesten Stand gebracht und steht seit August 2013 wieder in sämtlichen bisherigen Übersetzungen zur Verfügung [aptitude-dokumentation].

Das kommt insbesondere Anwendern entgegen, die Dokumentation lieber online lesen (oder „ergooglen“) statt sich die (stets aktuellen) Dokumentationspakete aus den Repositories auf ihrem System zu installieren. Ausführlicher gehen wir auf das Thema in Kapitel 9 ein.

Bei unserer Arbeit am Buch entdeckten wir zahlreiche Lücken in den Programmbeschreibungen, den Manpages und den beigefügten, weiterführenden Dokumentationen [bugs-found-during-book-writing]. Dabei wurde uns auch bewusst, welche Bedeutung dem persönlichen Erfahrungsschatz und insbesondere dem passiven Wissen zukommt. Wir haben uns bemüht, davon möglichst viel in dieses Buch einfließen zu lassen.

Dokumentation deb vs. rpm

Trotz vieler Fortschritte sind manche Programme zur Paketverwaltung und Hinweise zum Zusammenspiel von dpkg, APT und aptitude nur bruchstückhaft oder gar nicht beschrieben – oder sie sind über viele Köpfe und Online-Ressourcen hinweg verstreut. Auch an Übersetzungen mangelt es: So liegt trotz des hohen Nutzungsgrades beispielsweise die aptitude-Dokumentation bisher nicht in deutscher Sprache vor.

Im Vergleich steht das Paketformat RPM etwas besser da. In seinem Buch „Maximum RPM“ [Bailey-Maximum-RPM] hat Edward C. Bailey im Jahr 2000 die Regieanweisungen für den Umgang mit diesem Format veröffentlicht. Aktueller sind der „RPM Guide“ des Fedora-Projekts [Foster-Johnson-RPM-Guide] und weiterführende Dokumentationen auf der rpm-Projektseite [RPM-Webseite].

Ein vergleichbares Buch zur Debian-basierten Paketverwaltung fehlte bislang. Viele hervorragende Kompendien (siehe dazu Abschnitt 9.7) behandeln zwar die einzelnen Kommandozeilenwerkzeuge dpkg, APT, aptitude oder Synaptic, aber meist fehlt der (entscheidende) Entwurf eines Gesamtbildes, das sich erst aus der geschickten Kombination dieser Werkzeuge ergibt.

Was ist das Buch – und was nicht ...

Wir stellen dpkg, APT und aptitude mit den zugrundeliegenden Mechanismen in den Mittelpunkt. Wir erläutern die Unterschiede und ordnen die Werkzeuge anhand konkreter Aufgabenstellungen in den realen Einsatzkontext ein. Diesem problemorientierten Ansatz folgend, werden Sie die Programme künftig effizienter einsetzen und Paketmanagement als ebenso hilfreichen wie angenehmen Teil der Administration der Ihnen anvertrauten Systeme erleben.

Gedacht ist das Buch als Nachschlagewerk und Lernmedium für den Alltag. Es hilft Ihnen, (typische) Fehler oder Umwege zu vermeiden, und räumt mit zahlreichen Missverständnissen auf, die beim Thema Paketmanagement immer noch kursieren.

Unser Buch ist kein allgemeines Linux-Einsteiger-Buch in der Geschmacksrichtung „Debian GNU/Linux“, sondern widmet sich mit der Paketverwaltung bei Debian-Systemen einem speziellen Teilaspekt der Systembetreuung. Folglich spielen andere Paketformate als deb allenfalls eine Nebenrolle (siehe Abschnitt 2.2). Andere Debian-Derivate (siehe Abschnitt 1.5) und Linux-Distributionen haben vieles von Debian GNU/Linux übernommen, und die Rezepte lassen sich daher oft in gleicher Weise anwenden. Wir können jedoch nicht garantieren, dass wirklich alle Ausführungen uneingeschränkt für andere Distributionen gelten. Sofern uns gravierende Abweichungen vom Debian-Standard bekannt sind, benennen wir diese und erklären, wie Sie in einem solchen Fall am besten verfahren.

Weiterhin ist dieses Werk kein Entwicklerhandbuch, aus dem Sie erfahren, wie Sie deb-Pakete bauen und diese in Debian einbringen. Dieses Thema würde den Rahmen des vorliegenden Werkes um ein Mehrfaches sprengen und bleibt daher außen vor.

Zielgruppe und Lernziele

Dieses Buch richtet sich in erster Linie an *Systemadministratoren* und „Gehäusedeckelabschrauber“[2]. Richtig sind hier Verwalter und Betreuer Debian-basierter Infrastrukturen sowie Fortgeschrittene, die eine solche Funktion anstreben. Ihnen dienen Teil 1 (*Konzepte*) und 2 (*Werkzeuge*) mit den darin beschriebenen Optionen als Nachschlagewerk. Teil 3 (*Praxis*) hingegen nutzen sie als Arbeits- und Planungsmittel zur bestmöglichen Nutzung der beschriebenen Werkzeuge im Alltag.

[2] Dieter Thalmayr in: Oberflächliches – Enlightenment als Alternative zu Gnome und KDE, Vortrag im Rahmen des 11. Linux-Infotages Augsburg, 24. März 2012

Für *Anwender*, die den Linux-Einstieg mit Ubuntu oder Linux Mint bereits erfolgreich absolviert haben und nun der Systemverwaltung jenseits graphischer Oberflächen entgegenfiebern, bilden die Teile 1 und 2 das unverzichtbare Handwerkszeug. Teil 3 entspricht der Kür fortgeschrittener Kenntnisse. Die Lernkurve wird für sie deutlicher steiler ausfallen, aber stets beherrschbar sein.

Vorkenntnisse

Der Umgang mit der *Kommandozeile* sollte Ihnen vertraut sein. Wir legen uns nicht auf eine bestimmte Shell oder eine Terminalemulation fest. Alle Beispiele wurden unter `bash` getestet, funktionieren aber auch unter anderen Shells, wie z.B. der `zsh` (Axel nutzt auf einigen seiner Systeme die `zsh` als Login-Shell für den Benutzer `root`, wie es auch auf der Linux-Live-CD Grml gehandhabt wird). Die von uns ausgewählten und hier abgedruckten Ausgaben im Terminal sind unabhängig von der verwendeten Shell.

Graphische Werkzeuge spielen hier nur eine untergeordnete Rolle. Sie kommen nur dann zum Einsatz, wenn etwas nicht anders möglich ist oder es um genau deren Besonderheiten geht. Wir gehen davon aus, dass Sie auf einem Serversystem arbeiten und dieses ggf. sogar aus der Ferne betreuen. In dieser Konstellation bilden graphische Werkzeuge die absolute Ausnahme.

Für Teil 1 (*Konzepte*) ist Linux-Grundwissen unabdingbar: neben der Arbeit auf der Kommandozeile also auch grundlegende Kenntnisse über den *Filesystem Hierarchy Standard* (FHS), der die Struktur der Hauptverzeichnisse und deren Inhalte definiert (siehe dazu [FHS-Linux-Foundation] und [Debian-Wiki-FHS]).

Teil 2 (*Werkzeuge*) bespricht neben Strukturen zur Paketverwaltung alle Paketoperationen im Alltag und setzt dafür zumindest das Wissen aus Teil 1 voraus. Um manche Beispiele oder vorgestellte Konzepte leichter nachvollziehen zu können, ist mehrjährige Erfahrung mit Linux oder als UNIX-Systemadministrator von Nutzen.

Teil 3 (*Praxis*) beleuchtet ausschließlich konkrete, komplexere Anwendungsfälle aus dem Alltag. Voraussetzung dafür ist eine Vertrautheit mit den Werkzeugen zur Paketverwaltung, da es in diesem Abschnitt „ans Eingemachte“ geht.

Hilfreich sind darüber hinaus *Englischkenntnisse*: Viele Bildschirmausgaben sind englisch, nicht zuletzt weil die Lokalisierung der einzelnen Pakete bislang unvollständig ist.

Sie müssen auf Ihrem System über *administrative Benutzerrechte* verfügen, um manche Beispiele nachvollziehen zu können. Wir weisen nicht jedes Mal explizit darauf hin[3]. In den Beispielen für die Kommandozeile erkennen Sie anhand des Prompt-Zeichens, ob administrative Rechte notwendig sind oder nicht (`#` falls ja, und `$`, falls nicht). Auf Ausnahmen weisen wir Sie an der betreffenden Stelle explizit hin.

Auch wenn `dpkg`, APT und `aptitude` stabil und zuverlässig funktionieren – gerade in der Rolle und mit den Berechtigungen eines Administrators können falsche Befehle viel kaputt machen. Wir empfehlen Ihnen darum, die vorgestellten Beispiele zunächst auf einem separaten *Testsystem* auszuprobieren – sei dies ein eigener Rechner, eine virtuelle Maschine oder auch nur eine `chroot`-Umgebung [Debian-Wiki-chroot].

Dabei spielt es kaum eine Rolle, welches APT-basierte System Sie verwenden. Die meisten Beispiele im Buch stammen von einem Debian 7 *Wheezy* (*oldstable*) oder einem Debian 8 *Jessie* (*stable*). Alle Ausnahmen sind entsprechend gekennzeichnet.

Und das können Sie nach der Lektüre ...

Haben Sie das Buch gelesen und die Beispiele am Rechner nachvollzogen, verfügen Sie über profunde Kenntnisse in der Paketverwaltung unter Debian GNU/Linux. Dazu gehört:

- Debian-Pakete sauber verwalten
- kleinere und mittlere Debian-basierte Infrastrukturen pflegen
- die richtigen Werkzeuge für die Pflege benutzen und mit der Paketverwaltung sowie den Werkzeugen effektiv umgehen
- nicht nur die Software verwenden, sondern auch wissen, *warum* etwas funktioniert
- Pakete und Software nach Wunschkriterien finden

[3] Sie erlangen diese je nach Konfiguration Ihres Systems über die Kommandos `su` oder `sudo` – oder indem Sie sich als Benutzer `root` auf Ihrem System anmelden.

- alternative Auflösungen für Paketabhängigkeiten finden, verstehen und anwenden

All dies qualifiziert Sie für das entsprechende Lernziel der LPI-Prüfungen. Darüber hinaus schaffen Sie sich damit die Grundlagen, um später eigene und fremde Pakete zu bauen und die Paketierung für Debian durchzuführen. Das ist zugleich eine Voraussetzung, um später als Debian-Paket-Maintainer agieren zu können [Debian-Wiki-Debian-Entwickler].

Buchinfo

Wir pflegen eine buchbegleitende Webseite unter der URL:

http://www.debian-paketmanagement.de/

Darauf finden Sie neben einer Liste der Errata und deren Korrekturen auch inhaltliche Ergänzungen und Aktualisierungen. Natürlich freuen wir uns auch über Ihre Fragen und Anmerkungen!

Danksagung

Eine Reihe von Menschen haben uns bei der Realisierung dieses Buches direkt oder indirekt unterstützt, sei es in Form von Anregungen, Kritik, Vorschlägen zur Ergänzung oder Fach- und Verständnisfragen. Diesen Menschen gebürt unser Dank:

- Elmar Heeb (für `aptitude-robot` und viele interessante Diskussionen)
- Dirk Deimeke (für Tipps zum Autor-Werden) [Hackerfunk]
- Arne Wichmann (für das Diagramm der Vertrauenskette in Debian – unter GPL)
- Annette Kalbow (für inhaltliche Vorschläge mit `apt-file`, `dpkg -l` und `dpkg -L` sowie die graphische Umsetzung der Landkarte)
- Mechtilde Stehmann (für die Sprachkorrekuren und die Vorschläge für die FAQ)
- Marco Uhl (für die Idee zum FAQ-Eintrag über *Debian Snapshots* [Debian-Snapshots] bei Testing- vs. Produktiv-Umgebung)
- Werner Heuser (für die Installation und den Umgang auf Embedded und Mobile Devices)
- Claude Becker (für Ideen und Korrekturlesen rund um das Parsen von Debian-Versionsnummern und APT-Pinning)
- Christoph Berg (für Tipps und Tricks rund um `reprepro` und seine Erfahrungen mit `apt.postgresql.org`) [APT-Repo-PostgreSQL]
- Dr. Thomas Fricke (für Ergänzungen rund um die Verteilung von Paketen auf mehrere Maschinen)
- Jens Wilke (Konfigurationsmanagement)
- Martin Schütte (`reprepro`)
- Michael Vogt (für Erklärungen rund um APTs `mirror://` Methode und *gdebi*)
- David Kalnischkies (für viele Detailerklärungen – z.B. zur Parameterverarbeitung von `apt-get` – und die endlosen Diskussionen darüber, die dennoch meist irgendwann in Erleuchtung endeten)
- Albrecht Barthel (für die vielen Infos und Einblicke zum Univention Corporate Server, UCS)
- Martin *Venty* Ebnöther (für ein weiteres paar Augen und Ohren zum Thema Paketmanagement)
- Dr. Markus Wirtz für die lange Unterstützung und Hilfe, das Buch auf seinen Weg zu bringen, für den Klappentext sowie fürs Lektorat der Einleitung und dem erstem Kapitel.
- Oliver Rath für seine Vorschläge zur besseren Lesbarkeit von Programmcode
- Karsten Merker für viele kleine Korrekturen

- Wolfram Schneider für den Hinweis zu `dh-make-perl` als spezialisierte Variante von `checkinstall` sowie zum Aktualisieren von LTS-Versionen (siehe Kapitel 23)

Nicht zu vergessen sind die Probeleser, die sich durch unser Manuskript gekämpft haben: Arne Wichmann, Thomas Winde, Jana Pirat, Jörg Dölz, Hagen Sankowski und Eberhard Hofmann. Vielen Dank für Eure Mühe und Geduld!

Teil I

Konzepte

Kapitel 1

Willkommen im Linux-Dschungel!

1.1 Was ist Debian?

Je nach Kontext bezeichnet „Debian“ entweder

- das Debian-Projekt, also den Zusammenschluss von mittlerweile um die 1000 Entwicklern (*Debian Developers*, kurz: DD) weltweit, die das freie Betriebssystem gemeinsam entwickeln und veröffentlichen

oder

- das vom Debian-Projekt entwickelte Betriebssystem „Debian GNU/Linux“ bzw. dessen Varianten. Dazu zählen derzeit auch Debian GNU/kFreeBSD [Debian-Wiki-Debian-GNUkFreeBSD] und Debian GNU/Hurd [Debian-Wiki-Debian-GNUHurd], die statt eines Linux-Kerns einen FreeBSD- bzw. GNU-Hurd-Betriebssystemkern nutzen.

Einer der Eckpunkte des vom Debian-Projekt entwickelten Betriebssystems ist die ausschließliche Verwendung freier Software. Dafür sind die *Debian Free Software Guidelines* (DFSG) [DFSG] maßgeblich, die im *Debian-Gesellschaftsvertrag* festgelegt sind [Debian-Social-Contract]. Sichtbar wird das auch darin, dass Pakete in den beiden Entwicklungszweigen *contrib* und *non-free* offiziell kein Bestandteil von Debian GNU/Linux sind. Genauer gehen wir darauf in Abschnitt 2.9 ein.

Debian ist weder kommerziell noch profitorientiert. Das gesamte Projekt finanziert sich ausschließlich durch Spenden [Debian-Donations]. Dazu zählen nicht nur Geldspenden zufriedener Benutzer, sondern auch die Arbeitszeit von Entwicklern, Hardwarespenden oder das Betreiben eines Debian-Mirrors oder gar eines dedizierten Rechners für das Debian-Projekt.

Angestrebt wird ein universelles Betriebssystem, d.h. es gibt keinen Fokus auf einen spezifischen Einsatzbereich wie bei vielen Derivaten von Debian. Desweiteren werden dem Benutzer viele Entscheidungen selbst überlassen: Er muss – anders als z.B. in Ubuntu – wissen, was er möchte. Daher richtet sich Debian an zielorientierte, ambitionierte Einsteiger, Fortgeschrittene, Experten und Profis oder solche, die es wirklich werden wollen.

Debian stellt dafür ein ausgereiftes, stabiles und zuverlässiges Betriebssystem inklusive aller Software dar. Es ist ein Betriebssystem, das die Debian-Entwickler selbst benutzen wollen[1]. Daher unterstützt Debian viele verschiedene Architekturen und ermöglicht eine einheitliche Administration auf verschiedensten Plattformen (siehe Abschnitt 1.2). Ausführliches Testen und das Bereinigen von Fehlern hat Vorrang vor brandaktueller Software.

Aus diesen Grundsätzen folgen weitere Eigenschaften, die sich insbesondere im Einsatzzweck von Debian und der Einordnung in die Distributionsvielfalt widerspiegeln. Die typischen Anwendungsbereiche sind Server, Systeme für die Infrastruktur sowie Low-End Systeme wie etwa die Hardware-Lernplattform Raspberry Pi [RaspberryPi]. Dennoch hat sich Debian (nicht nur bei den Autoren) auch einen festen Platz auf dem Desktop erobert.

Zudem leiten sich aus Debian sehr viele Derivate für ausgewählte Zielgruppen oder Einsatzzwecke ab, z.B. Ubuntu, Linux Mint, Knoppix, Grml oder Damn Small Linux (DSL). Einen vollständigen Überblick („Stammbaum“) erhalten Sie in Abschnitt 1.5 sowie der GNU Linux Distribution Timeline [GNU-Linux-Distribution-Timeline].

[1] „The project consists of a group of people who are working together to create something that, primarily, we all want to use“ [Allbery-Debian-Popularity]

1.2 Debian-Architekturen

Debian kommt mit den unterschiedlichsten Hardware-Architekturen zurecht. Die offizielle Liste der aktuell unterstützten Architekturen finden Sie auf der Debian-Webseite [Debian-Architekturen] sowie im Anhang dieses Buches (siehe Abschnitt 45.1). Neben den veralteten Architekturen (siehe Abschnitt 45.2) werfen wir auch einen Blick in die Zukunft (siehe Abschnitt 45.3).

Nicht alle „Architekturen“ sind wirklich nur von der Hardware-Architektur abhängig, auf der die Programme einsetzbar sind, sondern auch von weiteren Punkten. Dazu zählen etwa der Betriebssystemkern, wie Linux, GNU Hurd [Hurd] oder FreeBSD [FreeBSD], aber auch die Art, wie die Programme kompiliert wurden (*Application Binary Interface*, ABI). Daher bezeichnen Entwickler dies als *Portierung* (*Port*) und sich selbst als *Porters*. Hier verwenden wir durchgängig den Begriff *Architektur*, da das entsprechende Feld in den Metadaten eines Pakets (siehe Kapitel 4) *architecture* heißt und Debian selbst die Begriffe bislang nicht konsistent verwendet.

Eine vollständige Liste der von `dpkg` verstandenen Architekturen gibt Ihnen der Aufruf `dpkg-architecture -L` im Terminal aus. Viele der in der Ausgabe des Kommandos genannten Architekturen existieren allerdings nur in der Theorie und zeigen auf, welche Möglichkeiten bestehen.

Architekturen, die das Werkzeug dpkg unterstützt (Ausschnitt)

```
$ dpkg-architecture -L
uclibc-linux-armel
uclibc-linux-alpha
uclibc-linux-amd64
m68k
sparc
sparc64
...
$
```

Die Übersicht im Anhang (siehe Kapitel 45) beschreibt die einzelnen Architekturen näher. Die verwendeten Bezeichnungen in Klammern geben dabei das entsprechende GNU-Triplet an, sofern dieses bekannt ist. Das GNU-Triplet besteht aus der Hardware-Plattform, dem Kernel und dem ABI.

Mit Hilfe des Perl-Moduls `Dpkg::Arch` ermitteln Sie diese Bezeichnungen im Handumdrehen selbst. Nachfolgend sehen Sie einen Aufruf für die Plattformen PPC64, PowerPC-spe, Arm, Armel und Armhf.

Perl-Aufruf zur Ermittlung der GNU-Triplets einer Debian-Architektur

```
$ perl \
    -MDpkg::Arch=debarch_to_gnutriplet \
    -E 'map { say "$_ = ".debarch_to_gnutriplet($_) } @ARGV' \
    ppc64 powerpcspe arm armel armhf

ppc64 = powerpc64-linux-gnu
powerpcspe = powerpc-linux-gnuspe
arm = arm-linux-gnu
armel = arm-linux-gnueabi
armhf = arm-linux-gnueabihf
$
```

1.2.1 Debian-Ports-Projekt

Das Debian-Ports-Projekt [Debian-Ports-Projekt] stellt die Infrastruktur für APT-Archive und automatisiertes Bauen von Paketen für Architekturen bereit, die Debian noch nicht oder nicht mehr unterstützt. Typischerweise gibt es dort nur zwei Kategorien von Veröffentlichungen: *unstable* und *unreleased*. Ersteres sind die gleichen Pakete wie in Debian *unstable*, nur wurden diese aus demselben Quellcode für diese spezifische Architektur übersetzt. Letzteres sind speziell für diese Architektur entwickelte oder modifizierte Pakete, die in den offiziellen APT-Archiven von Debian auch nicht im Quellcode zu finden sind.

In gewisser Weise stellt das Debian-Ports-Projekt dadurch gleichzeitig den Kreißsaal und das Altersheim für Debian-Architekturen dar – Anfang und Ende.

1.2.2 Pakete für alle Architekturen

Neben den bereits genannten Architekturen gibt es noch Pakete mit dem Eintrag *all*. Dies sind architekturunabhängige Pakete und Sie können diese auf beliebigen Architekturen installieren.

Dazu zählen z.B. Pakete von Programmen, die vollständig in den Skriptsprachen Perl, Python, Ruby oder Tcl geschrieben wurden. Ebenfalls gehören zu dieser Gruppe Pakete, die lediglich Daten enthalten, die auf jeder Architektur identisch sind. Das betrifft z.B. Bilder und Musik.

Auswahl der installierten, architektur-unabhängigen Pakete

```
$ dpkg -l | fgrep " all" | head -5
ii  abiword-common        3.0.0-8    all
    efficient, featureful word processor with collaboration -- common files
ii  acpi-support-base     0.142-6    all
    scripts for handling base ACPI events such as the power button
ii  adduser               3.113+nmu3 all
    add and remove users and groups
ii  adwaita-icon-theme    3.14.0-2   all
    default icon theme of GNOME
ii  aglfn                 1.7-3      all
    Adobe Glyph List For New Fonts
...
$
```

1.2.3 Multiarch: Mehrere Architekturen gleichzeitig auf einem System

Seit etwa 2004 läuft unter den Debian-Entwicklern die Diskussion um den Support für *multiarch* [Debian-Wiki-multiarch]. Unterstützung dafür gibt es in Debian seit Version 7 *Wheezy* und in Ubuntu seit Version 11.10 *Oneiric Ocelot*. Es beschreibt zwei Dinge:

- Systeme, auf denen Sie Pakete unterschiedlicher Architekturen nebeneinander benutzen können.
- Architekturspezifische Pakete, die explizit auf mehreren Architekturen installierbar sind.

Die Gründe für diese Mischung sind vielfältig:

- die Existenz von Systemen mit (nahezu) identischen Prozessorbefehlen (*Instruction Set*), aber unterschiedlicher Verarbeitungsbreite. Dazu zählen z.B. *i386*/*x86_64*, *ppc*/*ppc64*, *sparc*/*sparc64* und *s390*/*s390x*. Unterstützung hierfür gibt es bei RedHat/Fedora unter dem Namen *biarch* bereits länger [biarch].

  Dies ist insbesondere relevant bei proprietärer, nicht-quelloffener Software, die für 32-Bit-Linux kompiliert wurde, aber auf einem 64-Bit-System installiert bzw. verwendet werden soll.
- Systeme, die gemischte Prozessorbefehle unterstützen – entweder als Emulation in Hardware oder per Software. Dazu gehören z.B. *i386*/*ia64* mittels Hardware-Emulation und *arm*/jede Plattform (via Qemu Userland-Emulation).
- gemischte Betriebssystemumgebungen. Darunter fallen die Verwendung und Ausführung von Binärcode anderer Plattformen über eine Kompatibilitätsebene. Beispiele dafür sind Linux/*i386* auf FreeBSD/*i386* und Solaris/*sparc* auf Linux/*sparc*.
- Cross-Kompilieren. Darunter fällt das Übersetzen von Programmcode für eine andere Zielplattform.

Um diese Eigenschaft zu ermöglichen, bedarf es z.T. erheblicher Änderungen in den Übersetzungswerkzeugen und der Integration von Daten in der Dateistruktur. Dieser Vorgang ist bislang noch nicht vollständig abgeschlossen.

Benötigen Sie Pakete von einer anderen Architektur — bspw. ein *i386*-Paket (32 bit) auf einer *amd64*-Installation (64 bit) — ist diese parallele Installation und Benutzung der Software durchaus möglich. Wir zeigen Ihnen in Abschnitt 2.12, wie Sie diesen Schritt mit `dpkg` und `apt` erfolgreich bewerkstelligen.

1.2.4 Bevor es Multiarch gab

Wie oben bereits beschrieben, ist einer der Gründe hinter *multiarch* das Nutzen bereits kompilierter 32-Bit-Software auf 64-Bit-Systemen. Der Bedarf hierfür war auch schon vor der Entstehung von *multiarch* sehr groß.

Der Aufwand, alle üblicherweise genutzten *Shared Libraries* (zu dt.: gemeinsam genutzte Bibliotheken) der 32-Bit-Architektur *i386* zusätzlich auch noch als eigenes *amd64*-Binärpaket anzubieten, ist immens. Pakete dieser Form tragen üblicherweise das Präfix *ia32-* im Paketnamen. Vor der Entstehung von *multiarch* wurden daher *alle* notwendigen 32-Bit-Bibliotheken in ein einziges *amd64*-Binärpaket namens `ia32-libs` [Debian-Paket-ia32-libs] gepackt. Dieses Paket umfasste am Ende etwa stolze 800 MB und wurde in regelmäßigen Abständen mit den Sicherheitsaktualisierungen der entsprechenden Bibliotheken aufgefrischt.

Allein die Pflege dieses Pakets war schon recht mühsam. Ab der Einführung von *multiarch* wurde es gegenstandslos. Darum ist es seit Debian 7.0 *Wheezy* ein (leeres) Übergangspaket auf die passenden *multiarch*-fähigen Einzelpakete der Architektur *i386*.

1.3 Vom `tar.gz` zur Linux-Distribution

Der Begriff Linux-Distribution bezeichnet die Zusammenfassung von Softwarepaketen aus unterschiedlichen Quellen und deren gemeinsame Verteilung unter einem Distributionsnamen. Einen hohen Bekanntheitsgrad haben heute z.B. RedHat/Fedora, Debian, SuSE-Linux, Ubuntu, Knoppix und Linux Mint erreicht.

Die Vorteile einer Distribution liegen klar auf der Hand: aktuelle, stabile Versionen der Programme und insbesondere die Abstimmung der einzelnen Pakete aufeinander. Letzteres leistet der Distributor und nimmt damit Ihnen als Nutzer erhebliche Arbeit ab. Sie können sich darauf konzentrieren, die Distribution bzw. die Programme daraus zu verwenden.

Die ersten Linux-Distributionen entstanden zu Beginn der 1990er Jahre. Zu den Pionieren zählen Yggdrasil, SLS, Slackware, SuSE, RedHat und Debian. Bis dahin gab es kaum spezifische Pakete für jedes System – jeder Anwender passte die Software nach seinen eigenen Bedürfnissen an und pflegte diese Version dann kontinuierlich weiter. Zumeist waren das einfache `tar.gz`-Archive, die von Hand ergänzt und vorrangig für das eigene System übersetzt wurden.

Ein automatisiertes Verwalten der Software war zu diesem Zeitpunkt noch nicht möglich, weil die Strukturen nicht erdacht und umgesetzt waren. Abhängigkeiten der Software ließen sich nicht automatisch auflösen. Als Benutzer mussten Sie einerseits wissen, welche Software einander bedingte, und andererseits, welche Versionen und Varianten sich miteinander vertrugen. Namensgleiche Dateien und Verzeichnisse waren problematisch. Die große Kunst bestand im Wissen, in welcher Reihenfolge Sie zueinander passende Versionen von Software zunächst auswählen und diese nachfolgend auf Ihrem Linuxsystem installieren und konfigurieren mussten.

1.4 Debians Paketsystem

Aus diesen Erfahrungen heraus startete 1993 das Debian-Projekt unter Ian Murdock [Debian-History] mit einer revolutionären Idee: dem Bereitstellen kompilierter, vorkonfigurierter und sauber aufeinander abgestimmter Softwarepakete. Es folgte die Entwicklung von `dpkg` (*d* für *Debian* und *pkg* für *Package*), welches bis heute ein robuster Grundstein des Systems geblieben ist. Das verwendete `deb`-Paketformat und die dazugehörigen Werkzeuge wurden später von etlichen Linux-Distributionen übernommen. Ausführlicher beleuchten wir diesen Aspekt in Abschnitt 1.5.

Bald aber stieß das Werkzeug `dpkg` an Grenzen: Es installiert lediglich `deb`-Pakete, löst aber die Abhängigkeiten zwischen einzelnen Paketen nicht automatisch auf. Zudem muss das Paket bereits lokal vorliegen, d.h. `dpkg` kann es nicht direkt aus einem FTP- oder HTTP-Archiv beziehen.

Daraufhin begann die Entwicklung von `dselect`, welches aus dem Quellcode von `dpkg` gebaut wird, aber als eigenständiges Programm gilt. Später folgten `console-apt` (inzwischen aufgegeben) und `tasksel` (siehe Abschnitt 6.3.1), ab 1998 APT (*Advanced Packaging Tool*) sowie ab 1999 `aptitude` als Ncurses-basierte Oberfläche für `dpkg`. `dselect` wurde später weiterentwickelt und konnte somit auch APT als Backend benutzen.

Dabei lag die Zielrichtung auf der konsequenten Anwendung des UNIX-Prinzips „Ein Werkzeug für eine Aufgabe“. Das zeigt sich insbesondere darin, dass sich APT und `aptitude` an `dpkg` andocken und die verfügbaren Funktionen integrieren, indem die Programme bereits bestehende `dpkg`-Bibliotheken mitnutzen. Weitere Details dazu finden Sie in Abschnitt 2.3.

Heute stehen weitere textbasierte und graphische Benutzeroberflächen für `dpkg` zur Verfügung. Neben `aptitude` sind das Synaptic (siehe Abschnitt 6.4.1), PackageKit (siehe Abschnitt 6.4.5) – als Basis für Gnome-PackageKit und Apper bei KDE – sowie Muon (siehe Abschnitt 6.4.2), PackageSearch (siehe Abschnitt 13.4), Xara [Debian-Paket-xara-gtk] und SmartPM (siehe Abschnitt 6.4.3). Einen genaueren Blick werfen wir auf diese Programme in Kapitel 6.

1.5 Welche UNIX-artigen Betriebssysteme verwenden das Paketformat und das APT-Paketmanagement

Debian-Binärpakete liegen in einem spezifischen Format vor – dem `deb`-Paketformat. Sowohl das Format, als auch die dazugehörigen Werkzeuge haben innerhalb der letzten 20 Jahre bei weitaus mehr UNIX-artigen Betriebssystemen Einzug gehalten, als es auf den ersten Blick zu vermuten wäre.

Vereinfacht gesagt, basiert praktisch jedes Debian-Derivat auf den beiden Konzepten. Die Übersicht in Kapitel 47 zeigt eine Auswahl, jeweils ergänzt um den spezifischen Einsatzbereich. Bis auf den Univention Corporate Server (UCS) sind alle der genannten Derivate kostenfrei verfügbar.

Kapitel 2

Software in Paketen organisieren

2.1 Was ist Paketmanagement

Paketmanagement beschreibt die geordnete Verwaltung der einzelnen Softwarepakete auf ihrem System. Ziel ist dabei, dass Ihr Linux-System funktionstüchtig und benutzbar bleibt, insbesondere wenn Sie vorhandene Software aktualisieren, entfernen oder auch neue Software ergänzen.

Es umfasst daher nicht nur den Abgleich der lokalen Paketdatenbank mit den eingetragenen Paketverzeichnissen (*Repositories*), sondern auch die Auflistung der verfügbaren und derzeit verwendeten Pakete mit deren jeweiligen Statusinformation. Dazu gehört etwa die Paketbeschreibung, ob das Paket vollständig installiert ist und, falls ja, welche Version derzeit verwendet wird.

Weiterhin zählt zum Paketmanagement die automatische Auflösung von Paketabhängigkeiten. Das vereinfacht die Benutzung erheblich, da Sie die einzelnen Abhängigkeiten der Pakete nicht vorab recherchieren müssen. Diese Abhängigkeiten beeinflussen den lokalen Paketbestand und die Reihenfolge notwendiger Änderungen beim Hinzufügen, Aktualisieren oder Entfernen einer Paketauswahl. Daran schließen sich die plattform- und hardwarespezifische Konfiguration vor und nach der Installation von Paketen über die sogenannten Maintainer-Skripte an, die `dpkg` automatisch anstößt. Mehr Informationen dazu finden Sie in Abschnitt 4.2.

Die Distribution selbst bzw. die verantwortlichen Paketmaintainer kümmern sich bei der Übersetzung und Bereitstellung der Pakete darum, dass die nachfolgende Zusammenstellung der Paketliste harmonisch ist und die verschiedenen Versionen der einzelnen Softwarepakete aufeinander abgestimmt sind. Jedes `deb`-Paket verfügt über eine Beschreibung in Textform sowie eine Liste der Pakete, von denen es abhängt – bei Bedarf sogar samt Versionsangabe.

Die Aktualisierung einer bereits bestehenden, installierten Softwareversion durch eine andere Version beinhaltet i.d.R. eine fehlerbereinigte oder erweiterte Variante des Programms. Das kann eine individuelle Sicherheitsaktualisierung sein, das Installieren eines sogenannten *Debian Backports*, d.h. eine neuere Paketversion wird für eine vorherige Veröffentlichung zurückportiert, aber auch im Rahmen einer Aktualisierung auf eine neue Veröffentlichung der Distribution (siehe Abschnitt 2.10) stattfinden. Dass letzteres überhaupt möglich ist, ist noch lange nicht bei allen Distributionen selbstverständlich. Lange Zeit war dies ein Alleinstellungsmerkmal von Debian und auch heute noch bieten einige Debian-Derivate diese Eigenschaft nicht. Gleiches gilt für den Wechsel auf eine zurückliegende Softwareversion, einen sogenannten *Downgrade*. Dies wird allerdings auch bei Debian nicht explizit unterstützt, funktioniert aber dennoch in den meisten Fällen.

Im Detail erklären wir Ihnen die Thematik unter Pakete aktualisieren (siehe Abschnitt 8.39), Distribution aktualisieren (siehe Abschnitt 8.45), Paket downgraden (siehe Abschnitt 8.40) und dem Debian Backports Archiv (siehe Kapitel 19).

Nachfolgende Ausgaben zeigen zweierlei – die Liste aller Pakete am Beispiel von `dpkg` und die ausführliche Übersicht auf der Basis von `apt-cache`. Ersteres listet alle installierten Pakete zur Textverarbeitung Abiword auf. Ersichtlich ist der Installationsstatus (erste Spalte), der Paketname und die Paketversion (zweite und dritte Spalte) sowie eine Paketbeschreibung (vierte Spalte). Auf das Werkzeug `dpkg` gehen wir en detail in den beiden Abschnitten Softwarestapel und Ebenen (Abschnitt 2.3) und dpkg (Abschnitt 6.2.1) ein.

Ausgabe aller derzeit installierten Pakete für Abiword mit dpkg

```
$ dpkg -l "abiword*"
Gewünscht=Unbekannt/Installieren/R=Entfernen/P=Vollständig Löschen/Halten
```

```
| Status=Nicht/Installiert/Config/U=Entpackt/halb konFiguriert/
         Halb installiert/Trigger erWartet/Trigger anhängig
|/ Fehler?=(kein)/R=Neuinstallation notwendig (Status, Fehler: GROSS=schlecht)
||/ Name                Version        Architektur    Beschreibung
+++-==================-==============-==============-========================================== ←

ii  abiword             2.9.2+svn20120 i386           efficient, featureful word processor  ↩
    with co
ii  abiword-common      2.9.2+svn20120 all            efficient, featureful word processor  ↩
    with co
ii  abiword-plugin-gram 2.9.2+svn20120 i386           grammar checking plugin for AbiWord
ii  abiword-plugin-math 2.9.2+svn20120 i386           equation editor plugin for AbiWord
$
```

In Beispiel zwei nutzen wir `apt-cache` mit dem Parameter `showpkg`, um weitere Details zum Paket *abiword-common* zu erhalten. Neben der Versionsnummer sind auch die Paketquelle, die Paketsignaturen sowie die Abhängigkeiten zu weiteren Paketen genannt. Die Pakete stammen aus dem *main*-Zweig von Debian 7 *Wheezy*, sind für die Architektur i386 kompiliert und wurden vom deutschen FTP-Server des Debian-Projekts bezogen. Die einzige Abhängigkeit besteht zum Paket *abiword*.

Auflistung der Paketdetails zum Paket abiword-common mittels apt-cache

```
$ apt-cache showpkg abiword-common
Package: abiword-common
Versions:
2.9.2+svn20120603-8 (/var/lib/apt/lists/ftp.de.debian.org_debian_dists_wheezy_main_binary- ↩
    i386_Packages) (/var/lib/dpkg/status)
 Description Language:
                 File: /var/lib/apt/lists/ftp.de.debian.org_debian_dists_wheezy_main_binary ↩
                     -i386_Packages
                  MD5: 168081fc8391dc5eb8f29d63bb588273
 Description Language: de
                 File: /var/lib/apt/lists/ftp.de.debian. ↩
                     org_debian_dists_wheezy_main_i18n_Translation-de
                  MD5: 168081fc8391dc5eb8f29d63bb588273
 Description Language: en
                 File: /var/lib/apt/lists/ftp.de.debian. ↩
                     org_debian_dists_wheezy_main_i18n_Translation-en
                  MD5: 168081fc8391dc5eb8f29d63bb588273


Reverse Depends:
  abiword,abiword-common 2.9.2+svn20120603-8
Dependencies:
2.9.2+svn20120603-8 -
Provides:
2.9.2+svn20120603-8 -
Reverse Provides:
$
```

2.2 Varianten und Formate für Softwarepakete

Auf Linux-Systemen herrscht in Bezug auf das Paketformat keine Einheitlichkeit. Jede Linux-Distribution legt selbst fest, welches Paketformat sie verwendet. Zwei dieser Formate haben eine sehr hohe Verbreitung erlangt – `rpm` und `deb`. Slackware Linux nutzt hingegen ein schlichtes `tar`-Archiv, welches entweder mit `gzip` oder ab Release 13 mit `xz` komprimiert wird (siehe Tabelle 2.1).

Tabelle 2.1: Übersicht zu Paketformaten und deren Verbreitung

Abkürzung	Format	in Verwendung	Distribution
`deb`	Debian-Paketformat	seit 1993	Debian, Ubuntu, Grml, Knoppix, Linux Mint ...
`rpm`	Redhat Package Manager	seit 1995	RedHat/Fedora, CentOS, Mandrake/Mandriva/Mageia, SuSE/openSUSE, ...
`apk`	Android-Paketformat	seit 2003	Android
`ipkg`	Itsy Package Management System, Vorbild `deb`	2001 bis 2006	Unslung, OpenWrt, OpenMoko, webOS, Gumstix, iPAQ, QNAP (als Plugin), Synology (als Zusatz)
`opkg`	OpenMoko Package Management System, `ipkg`-Fork	seit 2006	OpenMoko, OpenWrt, OpenZaurus, OpenEmbedded
`pkg.tar.gz`	Pacman	seit 2002	Arch Linux
`tar.gz`, `tar.xz`	mit `gzip` bzw. `xz` komprimiertes `tar`-Archiv	seit 1993 (2009)	Slackware

Ändern Sie den Paketbestand auf Ihrem System durch eine Installation, Aktualisierung oder das Löschen eines oder mehrerer Pakete, ist in der Regel kein Neustart des gesamten Systems erforderlich. Die Paketpflege erfolgt bei laufendem System. Nach der Paketpflege ist üblicherweise lediglich der dazugehörige Dienst neu zu starten. Im Normalfall passiert dies heutzutage in den Maintainer-Skripten des Pakets und wird von der Paketverwaltung automatisch angestoßen. Mehr Informationen zu den Maintainer-Skripten finden Sie unter Aufbau und Format eines Debianpakets in Abschnitt 4.2.

2.3 Softwarestapel und Ebenen

2.3.1 Ebenen

Die Paketverwaltung kann man leicht in zwei Ebenen aufteilen. Dabei wird jede Ebene durch eine Reihe von Programmen und Bibliotheken repräsentiert (siehe Abbildung 2.1).

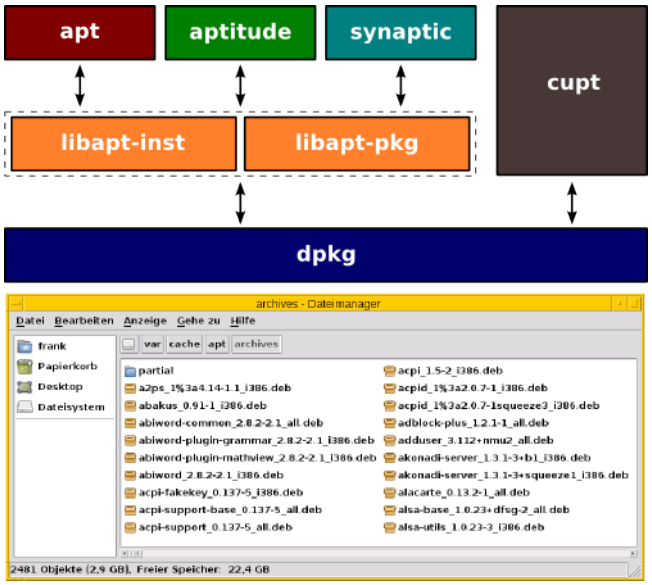

Abbildung 2.1: Schichtenmodell zur `deb`-basierten Paketverwaltung

2.3.2 Untere Ebene

Die Basis bildet `dpkg`. Dessen Aufgabe ist es a) ein bereits lokal vorliegendes `deb`-Paket auszupacken und auf dem System einzuspielen und b) die Inhalte eines bereits installierten `deb`-Pakets wieder aus dem System zu entfernen. Ersteres entspricht dabei dem Kommandozeilenaufruf `dpkg -i` *Paketdatei*, das zweite hingegen `dpkg -r` *Paketdatei* (siehe Abschnitt 8.36 und Abschnitt 8.41).

Für Statusabfragen zu einem einzelnen Paket stützt sich `dpkg` auf die beiden Hilfsprogramme `dpkg-deb` und `dpkg-query`. Dazu gehören bspw. die Schalter `-c` und `-L` zum Anzeigen des Inhalts eines Pakets (siehe Abschnitt 8.23) sowie `-l` zur Auflistung der installierten Pakete (siehe Abschnitt 8.5), `-s` zum Erfragen des Paketstaus (siehe Abschnitt 8.4) und `-S`, um das Paket zu finden, in dem eine bestimmte Datei vorkommt (siehe Abschnitt 8.22).

Mit `dpkg` können Sie Ihre Pakete verwalten und das System vollständig pflegen. Jedoch müssen Sie sich dann aber selbst um alle Komfortfunktionen kümmern. `dpkg` prüft nur, ob alle Abhängigkeiten zu anderen Paketen erfüllt sind und beendet im Fehlerfall die Aktion. Es nimmt Ihnen weder die automatische Auflösung von Paketabhängigkeiten, noch die richtige Reihenfolge bei der Installation der Pakete ab. Diese Mühe erleichtern Ihnen die Werkzeuge der oberen Ebene.

Paketverwaltung bei anderen Linux-Distributionen
Das Analogon zu `dpkg` bei `rpm`-basierten Distributionen ist `rpm`, bei Arch Linux ist es Pacman und bei Gentoo erreichen Sie die Funktionalität durch die beiden Programme `emerge` und `equery`. Eine komplette Übersicht zu den verschiedenen Programmen finden Sie einerseits in der Pacman-Rosetta (siehe Abschnitt 9.4) sowie in unserer Übersicht im Anhang des Buches (siehe Kapitel 46).

2.3.3 Obere Ebene

Bei `deb`-basierten Distributionen besteht die obere Ebene typischerweise aus dem Werkzeug APT (siehe Abschnitt 6.2.2). Häufig ist mindestens eines der weiteren Programme wie `aptitude` (siehe Abschnitt 6.3.2), Synaptic (siehe Abschnitt 6.4.1), Ubuntu Software Center (siehe Abschnitt 6.4.4), Muon (siehe Abschnitt 6.4.2) oder auch PackageKit (siehe Abschnitt 6.4.5) installiert. Die Auswahl variiert und hängt von der von Ihnen gewählten Linux-Distribution und ihren Vorlieben ab.

Alle diese Programme übernehmen die Aufgabe, Ihnen die Installation und die Aktualisierung der einzelnen Programmpakete auf Ihrem System zu vereinfachen und unter möglichst einer Benutzeroberfläche zusammenzufassen. Konkret gehört dazu die Aktualisierung der Liste von Paketen aus den Paketquellen, der Auflösung der Paketabhängigkeiten und die Berechnung der Installationsreihenfolge der von Ihnen ausgewählten Pakete.

Bei der Erfüllung ihrer Aufgaben stützen sich die Programme einerseits auf die beiden Bibliotheken `libapt-inst` und `libapt-pkg` (siehe Kapitel 5) und andererseits auf die Werkzeuge aus der unteren Ebene, d.h. vor allem auf `dpkg`. Es übernimmt die eigentliche Installation, Entfernung oder Aktualisierung (siehe untere Ebene). Sichtbar wird dies insbesondere, wenn Sie ein Paket mit `apt-get` oder `aptitude` installieren. Einen Teil der Ausgaben auf dem Terminal steuern `dpkg` und die o.g. Bibliotheken bei.

2.3.4 Paketformate und -werkzeuge anderer Distributionen

Bei `rpm`-basierten Distributionen RedHat, Fedora und CentOS heißen die Werkzeuge Yellowdog Updater, Modified (YUM) [YUM], bei SuSE und openSUSE Zypper [Zypper] und Yet another Setup Tool (YaST). Mageia Linux und Rosa Linux nutzen hingegen `urpmi` [Mageia-urpmi].

2.3.5 Werkzeuge, die verschiedene Paketformate unterstützen

Darüber hinaus gibt es Programme, die mit mehreren unterschiedlichen Paketformaten umgehen können. Dazu zählen Muon (siehe Abschnitt 6.4.2), der Smart Package Manager (siehe Abschnitt 6.4.3) und PackageKit (siehe Abschnitt 6.4.5). Muon und SmartPM können die Paketformate `deb`, `rpm` und `tar.gz` (Slackware) verarbeiten sowie die bereits oben genannten Verwaltungen APT, YUM und `urpmi` ansprechen. Weitere Informationen dazu finden Sie unter „Frontends für das Paketmanagement“ in Abschnitt 6.1.

2.4 Alternativen zu APT

APT mit `apt-get` und `apt-cache` ist erprobt, zuverlässig und daher weit verbreitet. Dennoch gibt es Programme, die die gleichen Funktionalitäten wie APT implementieren. Dabei gibt es verschiedene Kategorien von Alternativen:

Alternative Benutzerschnittstellen
: Hierzu zählen u.a. die im Buch vorgestellten Programme `aptitude`, Muon, Synaptic und wajig (siehe Abschnitt 6.3.2, Abschnitt 6.4.2, Abschnitt 6.4.1 und Abschnitt 6.2.4). Diese setzen auf den APT-Bibliotheken auf und sind nur Alternativen zu den Kommandozeilentools `apt-get` und `apt-cache`, nicht aber zu APT als Ganzes.

Vorgänger
: Bevor es APT gab und an Popularität gewann, wurden Paketlisten und Pakete mit `dselect` heruntergeladen (Recherchen ergaben etwa das Jahr 1998). `dselect` ist Bestandteil des `dpkg`-Projekts und wird heute noch aus den Quellen von `dpkg` gebaut. Allerdings benutzt es für viele Funktionalitäten mittlerweile ebenfalls APT als Backend, insbesondere für das Herunterladen von Paketen. `dselect` hat heute keine Relevanz mehr (liegt quasi im Wachkoma) und wird daher im Buch nicht weiter besprochen.

Potentielle Nachfolger
: APT ist nicht mehr ganz jung, und es wurden in der Vergangenheit Design-Entscheidungen getroffen, welche aus heutiger Sicht eher als weniger gelungen gelten, sich aber nicht mehr oder zumindest nur mit sehr viel Aufwand korrigieren lassen. Eugene V. Lyubimkin war einer der APT-Entwickler und hat sich aus o.g. Grund aus der APT-Entwicklung zurückgezogen und eine Re-Implementierung von APT namens Cupt [Debian-Wiki-cupt] geschrieben (siehe Abschnitt 6.2.5).

2.5 Zusammenspiel von dpkg und APT

Wie bisher gezeigt wurde, bauen `dpkg`, APT und Freunde aufeinander auf. Dabei gibt es eine Reihe von Bibliotheken und weiteren Programmen, die zur Nutzung dieser Werkzeuge ebenfalls notwendig sind.

APT hängt vor allem von der aus dem APT-Quellcode gebauten Bibliothek *libapt-pkg*, von *gnupg* und *debian-archive-keyring* ab. Die beiden letztgenannten Pakete werden für die Validierung von digitalen Signaturen benötigt (siehe Abschnitt 8.35).

`dpkg` ist ein sog. *essentielles* Paket (siehe Abschnitt 2.13), hat also eher wenig Abhängigkeiten. Die meisten davon sind selbst *essentielle* Pakete und müssen daher nicht namentlich als Abhängigkeit in den Metadaten des Pakets aufgeführt werden. Sie

tauchen daher nur unter bestimmten Umständen explizit in den Abhängigkeitslisten auf, z.B. wenn bestimmte Einschränkungen bzgl. der Version bestehen.

Bei `aptitude` und den meisten anderen Frontends ist dies anders, denn diese sind alle um eine ganze Spur komplexer. Bei `aptitude` kommt z.B. noch die Benutzeroberfläche auf der Basis von Ncurses [Ncurses] hinzu. Abbildung 2.2, Abbildung 2.3 und Abbildung 2.4 zeigen die minimalen Paketabhängigkeiten für APT, `dpkg` und `aptitude` in graphischer Form.

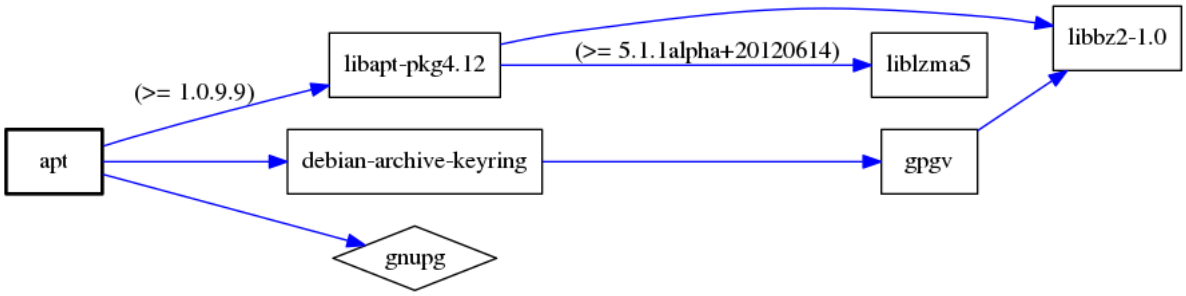

Abbildung 2.2: Paketabhängigkeiten von APT

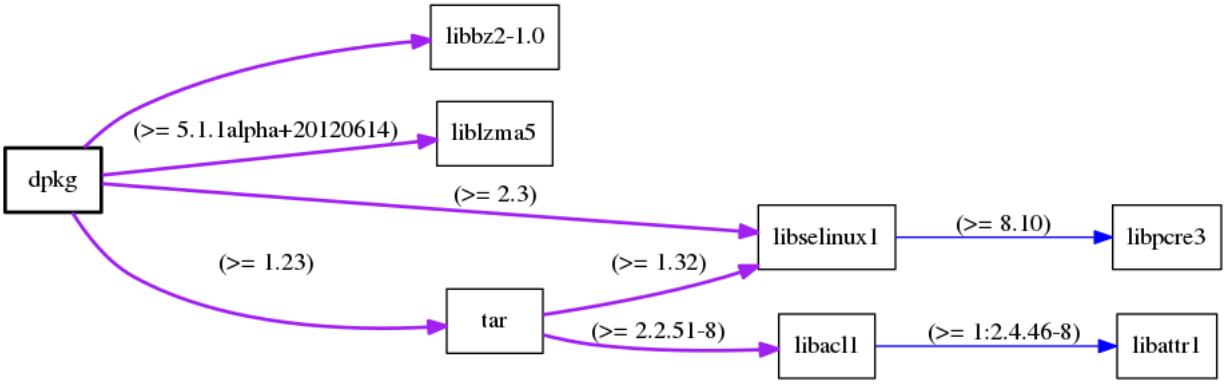

Abbildung 2.3: Paketabhängigkeiten von `dpkg`

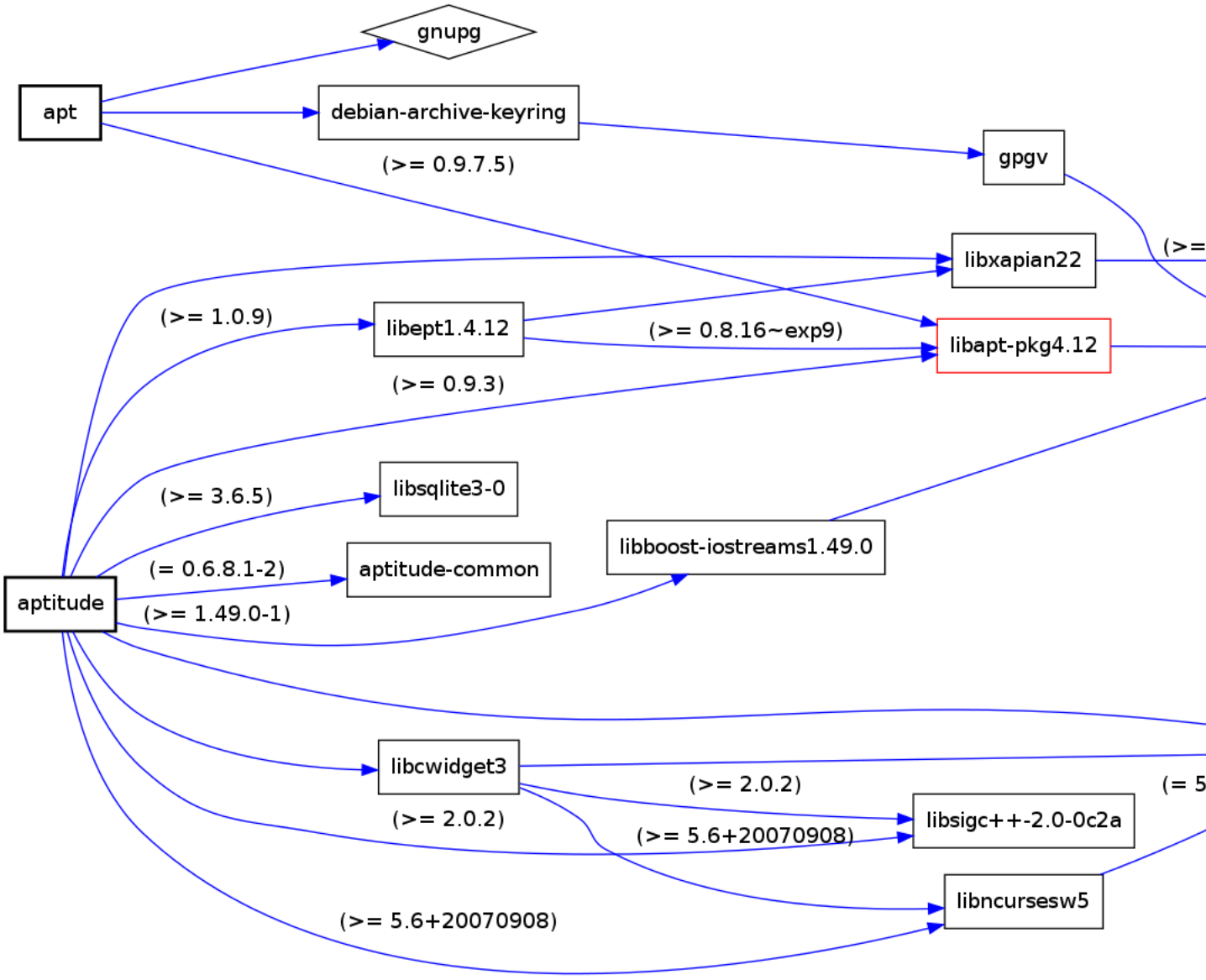

Abbildung 2.4: Paketabhängigkeiten von `aptitude` und APT

Die Grafiken in den drei obigen Abbildungen erzeugen Sie mit Hilfe der beiden Programme `debtree` [Debian-Paket-debtree] (siehe [debtree-Projektseite]) und `dot` [Graphviz]. Ersteres berechnet über die Metadaten in den Paketlisten die Abhängigkeiten zu anderen Paketen und erzeugt daraus eine entsprechende Beschreibung des Abhängigkeitsgraphen in der Sprache *dot*.

Erzeugung der Abhängigkeitsgraphen zu dpkg mittels debtree

```
$ debtree dpkg
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut
Statusinformationen werden eingelesen... Fertig
digraph "dpkg" {
        rankdir=LR;
        node [shape=box];
        "dpkg" -> "libbz2-1.0" [color=purple,style=bold];
        "dpkg" -> "liblzma5" [color=purple,style=bold,label="(>= 5.1.1alpha+20120614)"];
        "dpkg" -> "libselinux1" [color=purple,style=bold,label="(>= 2.3)"];
        "libselinux1" -> "libpcre3" [color=blue,label="(>= 8.10)"];
        "dpkg" -> "tar" [color=purple,style=bold,label="(>= 1.23)"];
```

```
        "tar" -> "libacl1" [color=purple,style=bold,label="(>= 2.2.51-8)"];
        "libacl1" -> "libattr1" [color=blue,label="(>= 1:2.4.46-8)"];
        "libacl1" -> "libacl1-kerberos4kth" [color=red];
        "tar" -> "libselinux1" [color=purple,style=bold,label="(>= 1.32)"];
        "dpkg" [style="setlinewidth(2)"]
        "libacl1-kerberos4kth" [style=filled,fillcolor=oldlace];
}
I: The following dependencies have been excluded from the graph (skipped):
I: libc6 multiarch-support zlib1g
// Excluded dependencies:
// libc6 multiarch-support zlib1g
// total size of all shown packages: 11501568
// download size of all shown packages: 4358750
$
```

Das zweite Kommando `dot` wandelt diese Beschreibung über den Schalter `-T`*Ausgabeformat* in eine hübsche Grafik um. Aus Ausgabeformat unterstützt `dot` derzeit bspw. PostScript, Structured Vector Graphics (SVG), GIF, PNG und FIG (für die Verwendung in `xfig`). Beachten Sie bitte, dass `dot` zwischen dem Schalter und dem von Ihnen gewählten Ausgabeformat kein Leerzeichen erlaubt.

Für `dpkg` erhalten Sie die Abbildung im Bildformat *Portable Network Graphics* (PNG) mit dem nachfolgend gezeigten Aufruf auf der Kommandozeile. Dabei wird die Ausgabe des `debtree`-Kommandos nicht auf dem Terminal sichtbar, sondern wird mit dem Pipe-Operator `|` direkt an das Programm `dot` weitergegeben, welches es als Eingabe verarbeitet. Die Ausgabe von `dot` – die erzeugte Bilddatei – wird danach mit dem Umleitungsoperator > in die Datei `dpkg.png` im aktuellen Verzeichnis umgeleitet.

Erzeugung der Abhängigkeitsgraphen und Speicherung als Rastergrafik

```
$ debtree dpkg | dot -Tpng > dpkg.png
$
```

2.6 Vom monolithischen Programm zu Programmkomponenten

Computerprogramme sind vergleichbar mit Kochrezepten und umfassen eine Folge von Anweisungen, die nacheinander abgearbeitet werden. Einfachere, kleine Programme sind häufig noch überschaubar und somit monolithisch. Sie beinhalten den gesamten Programmcode, der in einer einzigen Datei bereitgestellt wird.

Während zu Beginn der Informationsverarbeitung noch eine Tontafel, ein Holzstab mit Einkerbungen, ein Blatt Papier oder auch nur ein Lochstreifen zur Erfassung einer Folge von Anweisungen ausreichte, passt heutiger Programmcode nur noch selten auf eine einzige Bildschirmseite [Naumann-Abakus-Internet]. Ein Großteil der aktuell genutzten Software ist daher mehrteilig und überaus komplex. Dabei spielen viele, unterschiedliche Komponenten zusammen, erfüllen verschiedene Aufgaben und bedingen einander. Dazu gehören kompilierte Programme, Skripte, Bibliotheken, Daten und Konfigurationsdateien.

Die Paketierung der einzelnen Komponenten folgt eigenen Regeln, deren Konventionen nur zum Teil festgeschrieben sind und sich auch von Distribution zu Distribution etwas unterscheiden. Tabelle 2.2 zeigt die Zerlegung in einzelne Pakete am Beispiel von APT. Dabei beinhaltet die linke Spalte den generischen Paketnamen ohne Nennung von Versionsnummer und Architektur, die mittlere Spalte die Kategorie, der das Paket zugeordnet ist (siehe Abschnitt 2.8) und die rechte Spalte eine kurze Paketbeschreibung.

Tabelle 2.2: Paketierung der Komponenten am Beispiel von APT

Paketname	**Paketkategorie**	**Komponente und Bedeutung**
apt	Administration (*admin*)	Paketmanager für die Kommandozeile (siehe Abschnitt 6.2.2)
apt-doc	Dokumentation (*doc*)	Dokumentation zum Paket *apt*
apt-transport-https	Administration (*admin*)	APT-Plugin für HTTPS-Support
apt-utils	Administration (*admin*)	Hilfsprogramme für APT

Tabelle 2.2: (continued)

Paketname	Paketkategorie	Komponente und Bedeutung
libapt-inst	Bibliotheken (*libs*)	Laufzeitbibliothek zum Paketformat (siehe Kapitel 5)
libapt-pkg-dev	Bibliotheken zur Entwicklung (*libdevel*)	Entwicklerdateien zu *libapt-pkg* (siehe Kapitel 5)
libapt-pkg-doc	Dokumentation (*doc*)	Dokumentation zur Laufzeitbibliothek *libapt-pkg* (siehe Kapitel 5)
libapt-pkg	Bibliotheken (*libs*)	Laufzeitbibliothek zur Paketverwaltung (siehe Kapitel 5)

Benennung eines Debianpakets und Paketkategorien
In Abschnitt 2.11 beleuchten wir die Benennung und Abfolge der Komponenten in den Paketnamen. Eine genaue Auflistung und zur Bedeutung der Paketkategorien lesen Sie in Abschnitt 2.8 nach.

Die Ideen hinter der Zerlegung in einzelne Komponenten sind ganz unterschiedlich und ergeben sich aus der Entwicklung, dem Ausrollen und der nachfolgenden Pflege einer Software. Hauptmotivation ist dabei häufig, nicht das Rad jedes Mal neu erfinden zu müssen und stattdessen bereits bestehende Komponenten zu integrieren, die etabliert sind und bekanntermaßen einen gewissen Qualitätsstandard erfüllen. Im Open-Source-Bereich erfolgt die Entwicklung weltweit verteilt, daher ist hier eine Zerlegung in kleinere Einheiten und „Bausteine“ häufig von großem Nutzen. Aufgaben und Komponenten können dadurch besser an kleine, spezialisierte Teams verteilt werden.

2.7 Debian-Pakete (Varianten)

Wird von einem Debianpaket gesprochen, ist meist ein Binärpaket mit der Dateiendung `deb` gemeint. Dieses beinhaltet Software oder Daten, welche Sie sofort auf einem Computer mit Debian GNU/Linux installieren können.

Darüberhinaus gibt es aber auch noch andere Paketarten in Debian. Das wichtigste davon sind die Sourcepakete (siehe Abschnitt 2.7.4), die den Quellcode enthalten, aus dem später eines oder mehrere Binärpakete (siehe Abschnitt 2.7.1) gebaut werden.

2.7.1 Binärpakete (deb)

Binärpakete beinhalten Programme in kompilierter Form, die vorher bspw. in C oder einer ähnlichen Programmiersprache geschrieben wurden. Weiterhin beinhaltet es häufig noch Konfigurationsdateien, Dokumentation und weitere Daten in exakt dem Zustand, wie sie nachher auch auf der Festplatte Ihres Rechners vorliegen.

Bei der Installation eines `deb`-Pakets entpackt das Programm `dpkg` zuerst das Archiv aus dem `deb`-Paket und kopiert danach die Inhalte des Archivs an die vorbezeichnete Stelle in der Verzeichnishierarchie auf dem Zielsystem. Alle im Archiv genannten Pfade und Berechtigungen werden dabei übernommen.

Außerdem sind in den Binärpaketen Metadaten gespeichert, die solche Informationen wie bspw. die Abhängigkeiten zu anderen Paketen enthalten. Weitere Details dazu erfahren Sie unter „Konzepte und Ideen dahinter“ (siehe Abschnitt 4.1) sowie „Aufbau und Format von Binärpaketen“ (siehe Abschnitt 4.2.3).

Wie bereits oben benannt, hat ein Binärpaket üblicherweise die Dateiendung `deb` und wird auch durch das UNIX-Kommando `file` entsprechend als solches erkannt. Nachfolgende Ausgabe zeigt dieses Verhalten am Beispiel des Pakets *vnstat*, eines Programms zur Analyse des Netzwerktraffics.

Das UNIX-Kommando file identifiziert die deb-Datei als Debianpaket

```
$ file vnstat_1.10-1_i386.deb
vnstat_1.10-1_i386.deb: Debian binary package (format 2.0)
$
```

2.7.2 Übergangspakete, Metapakete und Tasks

Es gibt ein paar besondere Varianten von Binärpaketen – *Übergangspakete* und *Metapakete*. Vom Aufbau her unterscheiden sich diese nicht von normalen Binärpaketen, aber vom Inhalt.

Übergangspakete und Metapakete sind reguläre Binärpakete, die jedoch im Normalfall keine eigenen Programme, Daten oder ähnliches beinhalten. Stattdessen liefern diese Abhängigkeiten auf andere Pakete.

Übergangspakete werden bei Paketumbenennungen verwendet und dienen nur dazu, Ihnen den Wechsel bei geänderten (Binär-)Paketnamen zu erleichtern. Damit wird bei einer Aktualisierung eines bestehenden Pakets das Paket mit dem neuen Namen nachgezogen. In den meisten Fällen können Sie nach der Aktualisierung das Paket mit dem bisher verwendeten Namen gefahrlos von ihrem System entfernen. Nicht selten passiert dies bereits automatisch über die Paketverwaltung durch weitere, ggf. negative Abhängigkeiten.

Übergangspakete hängen meist nur von einem einzigen anderen Paket ab. Beispiele dafür sind:

- *git* → *gnuit* und dann später *git-core* → *git*
- *chromium* → *chromium-bsu* und dann später *chromium-browser* → *chromium*
- *diff* → *diffutils*
- *ttf-mplus* → *fonts-mplus*

Metapakete sind hingegen bewusst installierte Pakete, die Ihnen die Installation einer ganzen Gruppe von Paketen erleichtern. Als Abhängigkeiten zieht ein Metapaket eine Gruppe von verwendeten Paketen hinter sich her. Auf diese Art und Weise installieren Sie durch die Auswahl eines einzelnen Pakets eine ganze Gruppe an weiteren Paketen, die thematisch zusammengehören und sich häufig auch einander bedingen.

Das ist sehr nützlich, wenn Sie sich sicher sind, dass Sie eine bereits vorbereitete Zusammenstellung von Programmen benötigen. Für die Desktop-Umgebung XFCE genügt es beispielsweise, das dazugehörige Metapaket namens *xfce4* auszuwählen. Andere Programmzusammenstellungen wie *gnome* (GNOME-Window-Manager), *lxde* (LXDE-Window-Manager) und *kde-full* (K Desktop Environment) handhaben das ähnlich.

Sehr intensiv verwendet das Projekt Communtu diese Metapakete. Über die Webseite des Projekts stellen Sie sich individuelle Paketkombinationen („Bündel“) zusammen und beziehen diese von dort. Ausführlicher gehen wir darauf in Abschnitt 6.5.4 ein.

Tasks – auf deutsch mit „Aufgaben“ übersetzbar – sind Metapakete, die vom Debian Installer verwendet werden, um bestimmte Paketgruppen zu installieren. Dabei geht es vor allem um Pakete für bestimmte Sprachen und Lokalisierungen. Zum Beispiel hängt die Aufgabe *task-german-desktop* u.a. von den Paketen mit den deutschsprachigen Hilfedateien und Wörterbüchern von LibreOffice ab. Ähnliches existiert für Serverfunktionen, bspw. *task-dns-server* und *task-database-server*. Diese Funktionalität stammt vom Paket *tasksel* und wird ab Debian 7 *Wheezy* intensiv verwendet. Auf das angesprochene Programm `tasksel` gehen wir in Abschnitt 6.3.1 ausführlicher ein.

2.7.3 Mikro-Binärpakete

Ausschließlich die *Mikro-Binärpakete* mit der Dateiendung `udeb` sind technisch keine gewöhnlichen Binärpakete. Sie sind aufs Kleinste heruntergestutzte Pakete, die nur eine Art von Paketrelation namens „hängt ab von“ kennen, desweiteren keine Maintainer-Skripte beinhalten und auch sonst kaum Metainformationen mitführen. Einziger Einsatzzweck dieser Mikro-Debs[1] ist im Debian Installer während des Zeitpunkts der Installation. Deswegen gibt es auch nur solche Pakete als `udeb`-Variante, die vom Debian Installer selbst gebraucht werden. Darunter zählen bspw. Pakete mit den Programmen zum Anlegen von Dateisystemen.

Aufbau und Format von Übergangs- und Metapaketen
Mehr Informationen zum Aufbau dieser Pakete finden Sie unter Aufbau und Format von Übergangs- und Metapaketen in Abschnitt 4.2.4.

---

[1] das „u“ soll den griechischen Buchstaben Mu („μ“) darstellen

2.7.4 Source-Pakete (`dsc` und weitere Dateien)

Diese Pakete beinhalten den Quellcode von Programmen. Die Bestandteile sind:

- der Originalquellcode als ein oder mehrere komprimierte `tar`-Archive. Je nach verwendetem Komprimierungsverfahren lauten die Dateiendungen `orig.tar.gz`, `orig.tar.bz2` oder `orig.tar.xz`.
- die Änderungen vom Original zum Debianpaket als komprimierter Patch. Diese Dateien haben klassisch die Endung `diff.gz` und wurden mit `gzip` gepackt. Liegen die Änderungen wie bei moderneren Sourcepaketen als komprimiertes `tar`-Archiv vor, wird als Dateiendung `debian.tar.gz` oder `debian.tar.xz` genutzt. Bei Letzterem kommt anstatt von `gzip` das Komprimierungswerkzeug `xz` zum Einsatz.
- eine Datei mit den Metadaten (Größe, Hashsummen, etc.) über die vorhergenannten Dateien. Genutzt wird die Dateindung `dsc` als Abkürzung für *Debian Source Control*.

Alle diese genannten Dateien stellen in der Gesamtheit ein einzelnes Debian-Source-Paket dar und beinhalten den Upstream-Quellcode plus Paketierung.

Auspacken von Debian-Source-Paketen
Zum Auspacken von Debian-Source-Paketen benutzen Sie das Programm `dpkg-source` aus dem Paket *dpkg-dev*. Müssen Sie das Source-Paket vorher noch herunterladen, so nutzen Sie besser den Aufruf `apt-get source` *Paketname*, welcher das Source-Paket herunterlädt und danach direkt mit `dpkg-source` auspackt. Mehr Informationen zu Source-Paketen finden Sie unter „Aufbau und Format von Sourcepaketen“ in Abschnitt 4.2.2 und „Sourcepakete beziehen“ in Abschnitt 8.33.

2.7.5 Virtuelle Pakete

Reale Binärpakete können zusätzlich deklarieren, dass sie die Funktionalität eines weiteren Pakets ebenfalls bereitstellen. Existiert dieses weitere Paket nicht auch als reales Binärpaket, wird es als virtuelles Paket bezeichnet. Das gleiche virtuelle Paket kann hierbei von verschiedenen Binärpaketen zur Verfügung gestellt werden.

Andere Pakete können von einem solchen virtuellen Paket abhängen. Um diese Abhängigkeit zu erfüllen, genügt es, wenn ein Paket installiert ist, welches dieses virtuelle Paket bereitstellt.

In Debian gibt es bspw. die virtuellen Pakete *xserver*, *x-display-manager* und *x-window-manager*, die typische Komponenten des X-Window-Systems zusammenfassen. Abbildung 2.5 zeigt beispielhaft die Auswahl für das virtuelle Paket *x-display-manager* in `aptitude`. In der ersten Spalte der Darstellung kennzeichnet dazu der Buchstabe `v` neben dem Namen des virtuellen Pakets diese spezielle Variante.

Zur Auswahl aus dem Paket stehen u.a. der Displaymanager Slim (Paket *slim*), der Gnome Display Manager in Versionen 2 und 3 (Pakete *gdm* und *gdm3*), der KDE Display Manager (Paket *kdm*), der WINGs Display Manager und der ursprüngliche X Display-Manager (Paket *xdm*). Der Screenshot in Abbildung 2.5 stammt von einem Debian-System, auf welchem GDM3 installiert ist. Das erkennen Sie an der Hervorhebung durch fettgedruckten Text und der Markierung `i` für „Paket ist installiert“ in der ersten Spalte der Darstellung (siehe auch Abschnitt 6.2.1 für weitere Darstellungsvarianten).

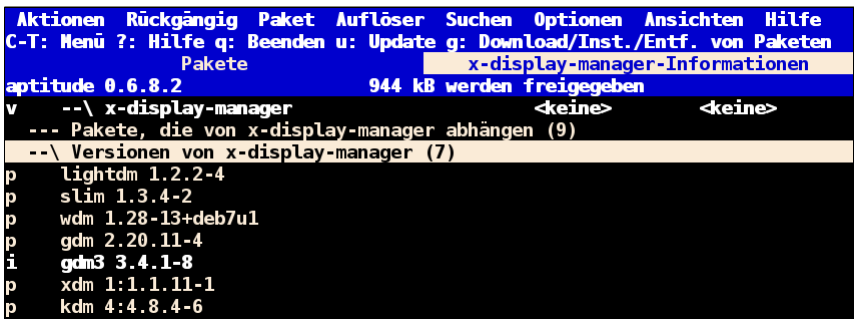

Abbildung 2.5: Inhalt des Pakets x-display-manager in Aptitude

Eine Liste aller offiziell verwendeten virtuellen Pakete in Debian gibt es im Paketierungshandbuch auf der Debian-Webseite [Debian-Virtual-Packages-List]. Andere Distributionen nutzen dieses Konzept auch, jedoch in unterschiedlicher Intensität.

2.7.6 Pseudopakete im Debian Bug Tracking System

Eine weitere Art nicht real existierender Pakete sind die sogenannten *Pseudopakete*, die Sie bei der Rückmeldung von Fehlern verwenden können. Diese Pakete dienen dazu, um Probleme mit der Debian-Infrastruktur aufzufangen und über das Debian Bug Tracking System (BTS) zu verfolgen.

Finden Sie bspw. einen Fehler auf den Webseiten von Debian, so können Sie einen Fehlerbericht gegen das Pseudopaket *www.debian.org* schreiben. Paketentfernungen aus Debian werden über Fehlerberichte gegen das Paket *ftp.debian.org* abgehandelt. Zukünftige Pakete sowie verwaiste Pakete werden über das Pseudopaket *wnpp* verwaltet und verfolgt. *wnpp* ist eine Abkürzung für „Work-needing and prospective packages“ — auf deutsch: „Arbeit bedürfende und zukünftige Pakete“.

Möchten Sie einen Fehlerbericht schreiben, wissen aber nicht, welchem konkreten Paket der Fehler zuzuordnen ist, so können Sie einen Fehlerbericht gegen das Pseudopaket *general* schreiben. Die Debian-Entwickler werden danach versuchen, herauszufinden, welches reale Paket die Ursache für den von Ihnen berichteten Fehler ist.

Fehler zu einem Paket anzeigen
Unter „Bugreports anzeigen“ in Abschnitt 32.3 lernen Sie, wie Sie die bestehenden Fehlermeldungen zu einem Paket anzeigen, deuten und einen eigenen Bugreport an das Betreuerteam des Pakets (*Paket-Maintainer*) übermitteln.

2.8 Sortierung der Pakete nach Verwendungszweck

Für Debian sind inzwischen sehr viele unterschiedliche Pakete verfügbar. Um Ihnen die Orientierung in der Paketmenge sowie die Recherche und Auswahl daraus zu erleichtern, ordnet der Paketbetreuer – der Verantwortliche für das Paket – dieses Paket *genau einer* bestimmten Kategorie zu. Die Auswahl der Kategorie basiert dabei auf dem hauptsächlichen Einsatzbereich der Software.

Abbildung 2.6 zeigt die Sichtbarkeit der Kategorien bei der Paketauswahl in `aptitude`. In jeder Kategorie sind die Pakete zusätzlich nach ihrem Distributionsbereich (siehe Abschnitt 2.9) – *main*, *contrib* und *non-free* – gruppiert. Der jeweilige Entwicklungszweig (siehe Abschnitt 2.10) – bspw. *stable*, *unstable* oder *testing* – wird in dieser Darstellung nicht angezeigt, lässt sich aber bei Bedarf als weitere Ebene in der Anzeigehierachie konfigurieren.

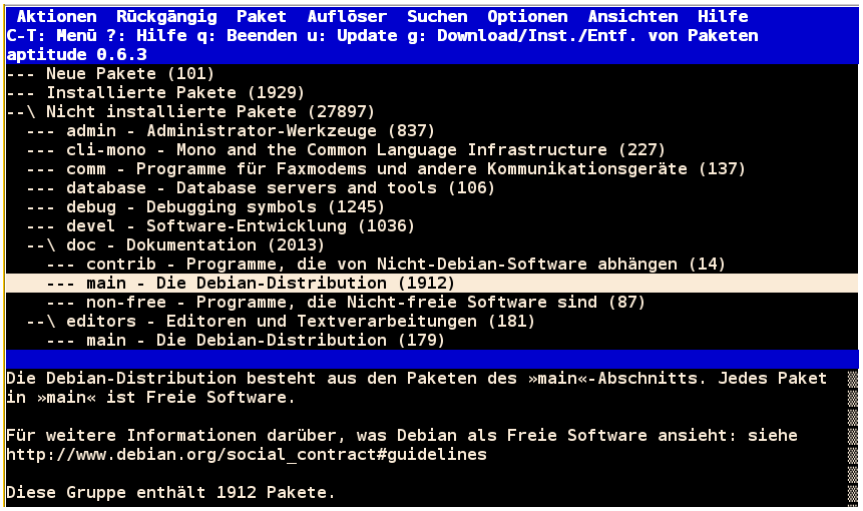

Abbildung 2.6: Auflistung der verschiedenen Paketkategorien in `aptitude`

Nachfolgende Übersicht listet die derzeit verwendeten Kategorien mit Beispielpaketen auf. Der Begriff in Klammern benennt die Kurzbezeichnung der Kategorie. Diese Zusammenstellung basiert auf Frank Ronneburgs Auflistung aus dem Debiananwenderhandbuch [Debian-Anwenderhandbuch] sowie der Übersicht auf der Debian-Webseite [Debian-Paketliste]. Die Kategorien *introspection*, *Debian/tasks*, *education* und *metapackages* sind derzeit noch nicht in allen Übersichten eingepflegt. Die einzige Referenz hierfür ist das Debian Policy Manual [Debian-Policy-Subsections].

Administration (*admin*)
Programme zur Systemadministration (*dpkg*, *apt*, *aptitude*, *adduser*)

Alte Bibliotheken und Übergangspakete (*oldlibs*)
Versionen von Bibliotheken, die nicht mehr verwendet werden sollten sowie Übergangspakete (*gcalctool*, *iproute*, *libgnome2-0*)

Amateurfunk/Ham Radio (*hamradio*)
Software für Amateurfunker (*ax25-tools*, *hamfax*)

Andere Betriebs- und Dateisysteme (*otherosfs*)
Software, um Programme zu benutzen, die für andere Betriebssysteme kompiliert wurden und um die Dateisysteme anderer Betriebssysteme zu benutzen (*avr-libc*, *bochs*, *cpmtools*, *dosemu*, *fatsort*)

Aufgaben (*Debian/tasks*)
Pakete, die Ihren Rechner für eine bestimmte Aufgabe vorbereiten (siehe Abschnitt 2.7) (*task-german-desktop*, *task-xfce-desktop*)

Bibliotheken (*libs*)
Programmbibliotheken (Libraries) (*libc6*, *e2fslibs*)

Bildung (*education*)
Lern- und Schulprogramme (*auto-multiple-choice*, *gcompris*, *scratch*)

Datenbanken (*database*)
Datenbankserver und -clients (*sqlite*, *mysql-server*, *mongodb*)

Debug-Pakete (*debug*)
Pakete, die Debug-Informationen für Programme und Laufzeitbibliotheken bereitstellen (*cups-dbg*, *evolution-data-server-dbg*)

Dienstprogramme (*utils*)
verschiedene Werkzeuge (*clamav*, *coreutils*, *debian-goodies*)

Dokumentation (*doc*)
HOWTOs, FAQs und andere Dokumentation sowie Programme, um diese zu lesen (*aptitude-doc-en*, *debian-faq*, *debian-handbook*, *zsh-doc*)

Editoren (*editors*)
Textverarbeitungsprogramme, Editoren für Programmierer und Entwickler (*abiword*, *emacs*, *kate*, *vim*)

Elektronik (*electronics*)
Programme zur Entwicklung und Simulation elektronischer Schaltungen (*arduino*, *verilog*)

Embedded (*embedded*)
Software, die für die Benutzung in oder mit Embedded Systemen geeignet ist (*gpe*, *matchbox*, *usbprog*, *urjtag*)

Entwicklung (*devel*)
Entwicklungswerkzeuge und -umgebungen, Compiler, usw. (*automake*, *binutils*, *g++*)

Entwicklungsbibliotheken (*libdevel*)
Header-Dateien zu Bibliotheken (*libc6-dev*, *okular-dev*, *zathura-dev*)

E-Mail (*mail*)
alles rund um E-Mail; Mailserver, Mailprogramme, Spamfilter, etc. (*postfix*, *mutt*, *spamassassin*)

GNOME (*gnome*)
Programme zur GNOME-Desktop-Umgebung (*etherape*, *evince*, *gnome-control-center*, *gnome-media*)

GNU R (*gnu-r*)
Programme um die freie Implementierung der Statistik-Sprache R (*r-base*, *r-mathlib*)

GNUstep (*gnustep*)
Programme zur GNUstep-Umgebung (*gnustep*, *gnustep-icons*)

Grafik (*graphics*)
Programme zur Bildbearbeitung (*dia*, *epub-utils*, *giftrans*, *gimp*)

Haskell (*haskell*)
alles rund um die Programmiersprache Haskell (*haskell-platform*, *happy*)

GObject Introspection (*introspection*)
GObject Introspection Middleware, Schnittstellen zwischen GObject-C-Bibliotheken und anderen Programmiersprachen [GObject-Introspection] (*gir1.2-ebook-1.2*)

Interpreter (*interpreters*)
Interpretierte Programmiersprachen wie bspw. Tcl/Tk (*luajit*, *m4*, *tcl*)

Java (*java*)
alles rund um die Programmiersprache Java (*ant*, *tomcat8*, *openjdk-7-jre*)

KDE (*kde*)
Programme zum KDE-Desktop (*apper*, *kdm*, *knotes*)

Kernel (*kernel*)
Betriebssystem-Kernel und zugehörige Module und Programme (*dkms*, *firmware-atheros*, *firmware-linux*, *kernel-package*, *linux-image-amd64*)

Klang (*sound*)
alles für den guten Ton (*alsa-utils*, *audacious*, *playmidi*, *xmms2*)

Kommunikation (*comm*)
Kommunikationsprogramme für externe Schnittstellen, Modems und Telefonanlagen (*cu*, *asterisk*, *hylafax-server*, *wvdial*)

Lisp (*lisp*)
alles zur Programmiersprache Lisp und Dialekten davon (*lush*, *mit-scheme*, *picolisp*)

Mathematik (*math*)
mathematische und wissenschaftliche Programme (*bc*, *concalc*, *euler*, *freemath*)

Metapakete (*metapackages*)
Paketgruppen (siehe Abschnitt 2.7) (*games-finest*, *gnome*, *kde-full*, *gis-devel*)

Mono/CLI (*cli-mono*)
alles rund um die C#-Implementierung Mono und die *Common Language Infrastructure* (*monodoc-browser*)

Netzwerk (*net*)
Netzwerkserver und Clientprogramme, Programme zur Netzwerkkonfiguration (*bind9*, *centerim*, *debmirror*, *isc-dhcp-client*)

Usenet News (*news*)
Software für Usenet-Newsgruppen (*slrn*, *nget*, *tin*)

OCaml (*ocaml*)
alles zur Programmiersprache OCaml (*cameleon*, *libcurl-ocaml*, *ocamlwc*)

Perl (*perl*)
alles zur Programmiersprache Perl, CPAN-Module (*libaudio-file-perl*, *perl*, *perl-doc*)

PHP (*php*)
alles zur Programmiersprache PHP (*icinga-web*, *php5*)

Python (*python*)
alles zur Programmiersprache Python (*python3*, *idle*)

Ruby (*ruby*)
alles zur Programmiersprache Ruby (*ruby*, *ruby-xmmsclient*)

Schriften (*fonts*)
Schriften und Programme zum Verarbeiten von Schriften (*fontforge*, *fontconfig*, *xfonts-cyrillic*)

Shells (*shells*)
verschiedene Shells (*bash*, *fish*, *zsh*)

Spiele (*games*)
Spiele und Unterhaltung (*freeciv-server*, *gcompris*, *openttd*)

Sprachpakete (*localization*)
Lokalisierungsunterstützung für große Softwarepakete (*iceweasel-l10n-all*, *kde-l10n-es*, *libreoffice-l10n-ar*)

TeX (*tex*)
alles zum Schriftsatzsystem TeX, inkl. LaTeX und XeTeX (*dvi2ps*, *biblatex*, *gummi*)

Textverarbeitung (*text*)
Werkzeuge zum Umgang mit Textdateien (*a2ps*, *xpdf*, *wordnet*, *wogerman*)

udeb-Pakete des Debian-Installers (*debian-installer*)
spezielle Pakete zur Verwendung im Debian-Installer, siehe Abschnitt 2.7.3 (*archdetect*, *cdrom-detect*)

Verschiedenes (*misc*)
Diverses, was sonst nirgends hineinpaßt (*bochsbios*, *cpuburn*, *screen*)

Versionskontrollsysteme (*vcs*)
Versionskontrollsysteme und zugehörige Hilfswerkzeuge (*bzr*, *cvs*, *git*)

Video (*video*)
Videobetrachter, -editoren, -rekorder, -sender (*dvb-apps*, *dvbstream*, *gnome-mplayer*, *mpv*)

Web (*web*)
Webbrowser, Download-Tools, HTML-Editoren, usw. (*bluefish*, *iceweasel*)

Webserver (*httpd*)
Webserver und ihre Module (*apache2*, *nginx*, *lighttpd*, *libapache2-mod-perl2*, *libapache2-mod-php5*)

Wissenschaft (*science*)
Programme zum wissenschaftlichen Arbeiten (*celestia*, *garlic*)

X Window (*x11*)
X-Server, Window-Manager und Anderes (*xterm*, *xsensors*, *xorg-xserver*)

XFCE (*xfce*)
Programme zum XFCE-Desktop (*thunar*, *xfwm4*, *xfwm4-themes*)

Zope/Plone (*zope*)
alles rund um das Zope-Framework (*zope-common*, *zope2.13*)

Erweiterung um Debtags
Das Kategorienkonzept hat eine Reihe von Limitierungen, insbesondere die Einordnung eines Pakets in nur eine einzige Kategorie. Um diese Grenzen aufzuheben, gibt es das Debtags-Projekt, welches jedes Paket um passende Schlagworte ergänzt. Dieses Konzept und die dazugehörigen Werkzeuge stehen unter „Erweiterte Paketklassifikation mit Debtags“ (siehe Kapitel 13) im Mittelpunkt.

2.9 Distributionsbereiche

Die verschiedenen Distributionsbereiche ordnen die einzelnen Pakete anhand ihrer Lizenzen. Das hilft Ihnen dabei, die Kontrolle über die verwendeten Lizenzen auf Ihrem System zu behalten. Mit der Auswahl von Paketen aus bestimmten Distributionsbereichen legen Sie somit den „Freiheitsgrad“ Ihrer Installation fest.

In Debian sind die Softwarepakete in die folgenden drei Bereiche unterteilt:

main
: Freie Software, die den Debian-Richtlinien für freie Software (DFSG) entspricht.

contrib
: Freie Software, die von unfreier Software abhängt.

non-free
: Software, die *nicht* den Debian-Richtlinien für freie Software (DFSG) entspricht, aber frei verteilbar ist.

2.9.1 Einordnung der Distributionsbereiche in Debian

Obwohl vielfach von außen anders wahrgenommen, zählt zur Debian-Distribution nur der Bereich *main*. Die anderen beiden Bereiche sind lediglich Ergänzungen, die zusätzlich bereitgestellt werden. Wir empfehlen Ihnen daher, soweit möglich nur Pakete aus *main* zu verwenden, und nur wenn dies nicht ausreicht (z.B. wegen nicht-freier Firmware für bestimmte Hardwarekomponenten), die beiden anderen Bereiche *contrib* und/oder *non-free* dazuzunehmen.

Pakete, die in *main* eingeordnet sind, unterliegen einer Lizenz, die den Debian-Richtlinien für Freie Software – kurz *Debian Free Software Guidelines (DFSG)* [DFSG] – entsprechen. Diese Richtlinien sind im Debian-Gesellschaftsvertrag festgelegt [Debian-Social-Contract] und weisen starke inhaltliche Gemeinsamkeiten zur Free Software Foundation (FSF) und zum GNU-Projekt auf.

Pakete im Bereich *contrib* stehen zwar genauso unter einer freien Lizenz wie die Pakete in *main*, bedingen jedoch weitere Software oder Inhalte, die nicht frei gemäß obiger Festlegung ist. Typische Gründe, warum ein Paket im Bereich *contrib* einsortiert wurde, sind:

- Eine freie Spiele-Engine braucht die Spieldaten eines kommerziellen Spiels.
- Ein Emulator braucht Software für die zu emulierende Hardware, um zu funktionieren.
- Die Software ist nur zum Herunterladen (und ggf. installieren und/oder paketieren) von nicht-freier Software da.
- Die Software muss mit einem nicht-freien Compiler übersetzt werden.

Im Bereich *non-free* finden sich Pakete, die nicht den Debian-Richtlinien für Freie Software (DFSG) entsprechen, aber trotzdem immer noch frei verteilbar sind. Typische Gründe für die Nichterfüllung der DFSG im Bereich *non-free* sind:

- Der Quellcode liegt nicht (komplett) vor.
- Die Software oder einzelne Teile davon – z.B. Teile der Dokumentation – dürfen nicht modifiziert werden.
- Die Software darf nur für nicht-kommerzielle Zwecke genutzt werden.
- Die Software darf nur für „Gutes“ verwendet werden.
- Die Software darf nicht in kompilierter Form verteilt werden.

Vor der Nutzung von Software aus diesem Bereich ist es ratsam, immer erst anhand der Lizenz zu überprüfen, ob Sie diese Software überhaupt für Ihre gewünschten Zwecke einsetzen dürfen.

Für Software aus dem Bereich *non-free* gilt außerdem, dass keine Unterstützung seitens Debian für diese Pakete möglich ist. Das trifft insbesondere dann zu, wenn der Quellcode nicht veröffentlicht wurde, wie das bspw. bei der Firmware zu bestimmten WLAN-Chipsätzen der Fall ist.

Abbildung 2.7 zeigt die Paketliste in Aptitude mit einem unfreien Paket aus dem Bereich Netzwerk – `skype`. Im abgebildeten Fall wurde es zudem nicht aus einem offiziellen Debian-Repository heruntergeladen, sondern manuell auf dem System eingepflegt.

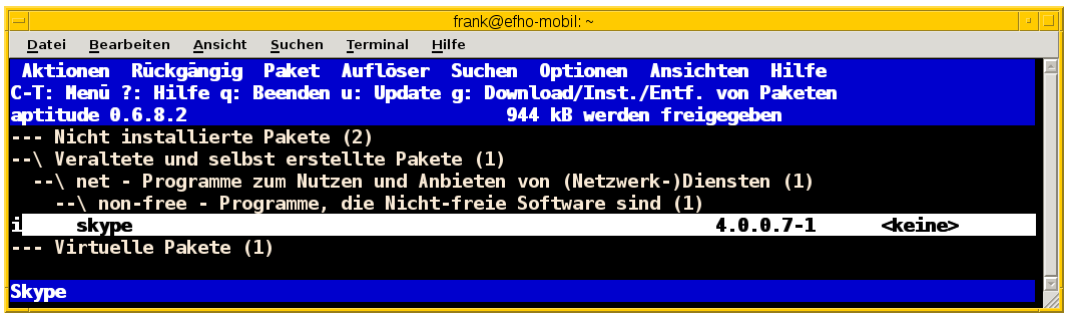

Abbildung 2.7: Paketliste mit Skype als unfreies Paket in Aptitude

Eine vollständige Übersicht zu allen nicht-freien Paketen, die auf ihrem System installiert sind, gibt Ihnen das Programm `vrms`. Darauf gehen wir unter „Liste der installierten, nicht-freien Pakete anzeigen“ (siehe Abschnitt 8.7) ausführlicher ein.

2.9.2 Einordnung der Distributionsbereiche bei anderen Distributionen

Bei Ubuntu sind die Distributionsbereiche etwas anders eingeteilt als bei Debian. Dort kommt neben den Lizenzen auch noch der Supportstatus zum Tragen. Dafür ist die Unterscheidung nach Softwarelizenzen auf frei oder unfrei reduziert: Es gibt *main* (frei, von Canonical unterstützt), *restricted* (unfrei, von Canonical unterstützt), *universe* (frei, nur Community-Unterstützung) und *multiverse* (unfrei, nur Community-Unterstützung).

Andere Derivate von Debian bzw. Ubuntu oder nicht-offizielle Paketquellen (siehe Abschnitt 3.1) können ebenfalls ihre eigenen Distributionsbereiche haben. Auf diese gehen wir hier nicht weiter ein.

2.9.3 Handhabung von geschützten Namen und Logos

Der Begriff „Software“ wird bei Debian recht weit gefasst und beinhaltet neben Programmcode auch Firmware, Dokumentation oder künstlerische Elemente wie beispielsweise Grafiken und Logos. Letztere stehen in manchen Fällen unter anderen Lizenzen als der Rest der Software und dürfen aus markenrechtlichen Gründen nicht für abgeänderte Programme verwendet werden. Aus diesem Grund wurden einige Programme abgewandelt, bspw. der Webbrowser Iceweasel und das Mailprogramm Icedove, die im Original die Namen Firefox und Thunderbird tragen. Neben den beiden anderen Namen werden in Debian auch alternative Logos genutzt.

2.9.4 Softwareverteilung

Bezogen auf die Anzahl der verfügbaren Softwarepakete findet sich der überwiegende Teil der Pakete im Bereich *main*, danach folgen *contrib* und *non-free*. Für die Architektur *amd64* in Debian 8 *Jessie* ist das Verhältnis 42987 (*main*) zu 250 (*contrib*) zu 470 (*non-free*). Damit sind das fast genau ein Prozent unfreie Pakete. Für die Plattform i386 ist die Verteilung ähnlich.

2.9.5 Hintergrund der Einteilung in Distributionsbereiche

In der Klassifikation spiegelt sich die Offenheit und Vielfalt der Debian-Nutzer und -Entwickler sowie deren Weltbild wieder. Es zeugt von dem Verständnis dahingehend, welche Software Sie tatsächlich verwenden und nach welchen Kriterien Sie Ihre Pakete auswählen.

Je mehr Nutzer von Debian einbezogen werden, umso vielschichtiger sind die Varianten der Verwendung. Jeder Nutzer pendelt sich bei der Paketauswahl irgendwo zwischen den beiden Polen „nur freie Software“ und „freie und unfreie Software gemischt“ ein.

Erstere Gruppe versucht, ausschließlich freie Software zu verwenden und dazu auch unfreie in freie Software zu überführen, bspw. durch Nachbau, Neuentwicklung oder Anregen eines Lizenzwechsels. Dieser Schritt kann auch mit einem Funktionsverzicht einhergehen und ist vergleichbar mit der Überzeugung „so lange eine Technologie nur kommerziell/unfrei zur Verfügung steht, verwende ich diese nicht und nutze stattdessen Alternativen“. Die zweite Gruppe ist deutlich pragmatischer und folgt dem Gedanken „ich nutze die unfreie Variante, bis eine freie zur Verfügung steht, und steige dann um, wenn sie das kann, wie ich es brauche“. Dazwischen gibt es unendlich viele Abstufungen, die wiederum persönlichen Schwankungen unterliegen können.

Die Nutzung der Software hängt von den Bedürfnissen und dem Einsatzzweck ab. Viele Prozesse und Arbeitsabläufe bedingen eine bestimmte Menge von Eigenschaften („Featureset“), welche sich nicht immer adäquat und vollständig mit bestehender freier Software bzw. deren aktuellem Entwicklungsstand abbilden lässt. Dabei spielen die Faktoren Produktivität, Anbindung an bereits bestehende Software, Schnittstellen und unterstützte Hardware oder Protokolle eine große Rolle. Desweiteren sind das Budget, der Zeitrahmen und die Dokumentation bzw. der Support entscheidend. Über die Auswahl einer Lösung entscheidet häufig, welcher finanzielle Rahmen für eine Lösung zur Verfügung steht, welcher Zeitraum zur Inbetriebnahme gesetzt ist und wie gut die Dokumentation und der Support zur ausgewählten Software ist. Eine Software, die frei ist, aber nicht oder nur ungenügend zum tatsächlichen Einsatzzweck passt, ist an dieser Stelle zu hinterfragen und muss sich mit einer passenden Alternative messen lassen, auch wenn diese als unfrei eingestuft ist, aber damit im Nutzungszeitraum eine funktionierende und stabile Lösung erreicht wird.

2.10 Veröffentlichungen

Debian GNU/Linux wird in verschiedenen Veröffentlichungen angeboten, die jeweils als „Releases“ bezeichnet werden. Eine solche Veröffentlichung kann wie folgt referenziert werden:

- nach ihrer Versionsnummer, z.B. *Debian 7* oder *Debian 8*
- nach dem aktuellen Entwicklungsstand der Veröffentlichung (siehe Abschnitt 2.10.1), z.B. *stable*, *testing* oder *unstable*
- nach ihrem Alias-Namen (siehe Abschnitt 2.10.2), z.B. *Wheezy* oder *Jessie*

2.10.1 Bedeutung der verschiedenen Entwicklungsstände

Jedes aktuelle Debian-Paket gehört zu mindestens einem der nachfolgend beschriebenen Entwicklungsstände:

unstable
: Hier findet die aktive Entwicklung von Debian statt. Neue Pakete und Versionen landen fast immer zuerst hier. Dieser Entwicklungzustand kann inkonsistent sein und beispielsweise unerfüllte Abhängigkeiten beinhalten. Er ist primär für Entwickler gedacht.

testing
: Pakete, die in *unstable* für eine gewisse Zeit keine schwerwiegenden Fehler aufweisen und deren Abhängigkeiten bereits ebenfalls in *testing* erfüllt werden können, wandern automatisch von *unstable* hierher. Dieser Entwicklungsstand sollte konsistent sein und alle Paketabhängigkeiten erfüllt sein.

stable
: Das ist die jeweils aktuelle stabile Veröffentlichung. Dieser Entwicklungsstand ist für die normale Nutzung von Debian empfohlen. Eine neue stabile Veröffentlichung wird ca. alle zwei Jahre auf Basis von *testing* erstellt. Pakete werden nur aktualisiert, um sicherheitskritische oder sonstige schwerwiegende Fehler zu beheben. Dabei werden (mit sehr wenigen Ausnahmen) ausschließlich die entsprechenden Fehler durch Patches behoben anstatt neuere Versionen der Programme auszuliefern.

oldstable
: Das ist die jeweils vorherige stabile Veröffentlichung. Bevor eine neue stabile Veröffentlichung freigegeben wird, erfolgt eine Umbenennung der aktuellen stabilen Veröffentlichung in *oldstable*. Diese wird von da an im Normalfall noch für ein Jahr weiter gepflegt und mit Sicherheitsaktualisierungen versehen.

  Im Frühjahr 2014 wurden zusätzlich sogenannte *Long Term Support*-Varianten – kurz *LTS* – eingeführt, die den Zeitraum der weiteren Verfügbarkeit und Pflege auf bis zu fünf Jahre verlängern. In Folge wird die im Jahr 2011 freigegebene und 2013 durch Debian 7 *Wheezy* abgelöste Veröffentlichung von Debian 6 *Squeeze* bis 2016 mit Aktualisierungen versorgt.

oldoldstable
: Wenn vorhanden, ist dies die jeweils vorvorherige stabile Veröffentlichung. Zum ersten Mal trat dieser Entwicklungsstand auf, als im Frühjahr 2015 Debian 8 *Jessie* zur stabilen Veröffentlichung erklärt wurde. Gleichzeitig wurde Debian 6 *Squeeze* zur neuen Suite *oldoldstable* und wird per *Long Term Support* weiterhin noch eingeschränkt unterstützt.

experimental
: Dies ist der einzige Entwicklungsstand, der keine alleinstehende Veröffentlichung ist, sondern nur ein Zusatz-Repository. Es fungiert als Erweiterung zu *unstable* und beinhaltet Pakete, bei denen der Paketbetreuer davon ausgeht, dass sie noch und ggf. sogar grobe Fehler beinhalten. *experimental* wird genutzt, um Pakete im größeren Umfeld zu testen, bevor diese nach *unstable* hochgeladen werden.

Darüberhinaus existiert der Paketbereich *backports*. Das beinhaltet Rückportierungen neuerer oder aktualisierter Pakete aus dem Entwicklungszweig *testing* nach *stable*, teilweise auch aus *unstable*. Das ist spannend, aber auch mit gewissen Risiken verbunden. Im Detail gehen wir darauf unter Debian Backports in Kapitel 19 ein.

2.10.2 Alias-Namen

Jede Veröffentlichung von Debian GNU/Linux hat einen Alias-Namen, der nach einer Figur aus Pixars Filmreihe Toy Story benannt ist. Für die bisherigen Veröffentlichungen standen die folgenden Figuren aus dem Film Pate:

- Debian 1.0 wurde nie offiziell veröffentlicht.
- Debian 1.1 *Buzz* (17. Juni 1996; benannt nach Buzz Lightyear, dem Astronauten)
- Debian 1.2 *Rex* (12. Dezember 1996; benannt nach dem Plastikdinosaurier)
- Debian 1.3 *Bo* (5. Juni 1997; benannt nach Bo Peep, der Schäferin)
- Debian 2.0 *Hamm* (24. Juli 1998; benannt nach dem Sparschwein)
- Debian 2.1 *Slink* (9. März 1999; benannt nach dem Hund Slinky Dog)
- Debian 2.2 *Potato* (15. August 2000; benannt nach der Puppe Mr. Potato Head)
- Debian 3.0 *Woody* (19. Juli 2002; benannt nach dem Cowboy Woody Pride, der Hauptfigur der Filme)
- Debian 3.1 *Sarge* (6. Juni 2005; benannt nach dem Feldwebel der grünen Plastiksoldaten)
- Debian 4.0 *Etch* (8. April 2007; benannt nach der Zeichentafel Etch-A-Sketch)
- Debian 5.0 *Lenny* (Februar 2009; benannt nach dem aufziehbaren Fernglas)
- Debian 6.0 *Squeeze* (Februar 2011; benannt nach den grünen dreiäugigen Aliens)
- Debian 7 *Wheezy* (Mai 2013; benannt nach Wheezy the Penguin, dem Gummi-Spielzeugpinguin mit der roten Fliege)
- Debian 8 *Jessie* (25. April 2015; benannt nach der jodelnden Kuhhirtinnen-Puppe Jessica Jane „Jessie“ Pride)
- Debian 9 *Stretch* (noch kein Veröffentlichungstermin zum Zeitpunkt der Drucklegung festgelegt, voraussichtlich 2017; benannt nach dem lila Kraken)
- Debian 10 *Buster* (noch kein Veröffentlichungstermin zum Zeitpunkt der Drucklegung festgelegt, voraussichtlich 2019; benannt nach dem Welpen aus *Toy Story 2*)

Mehr Details zu den einzelnen Veröffentlichungen finden sich in der Debian-Historie [Debian-History]. Die Figuren aus Toy Story und insbesondere deren Charakterzüge sind ausführlich im Disney Wiki [ToyStory] dokumentiert.

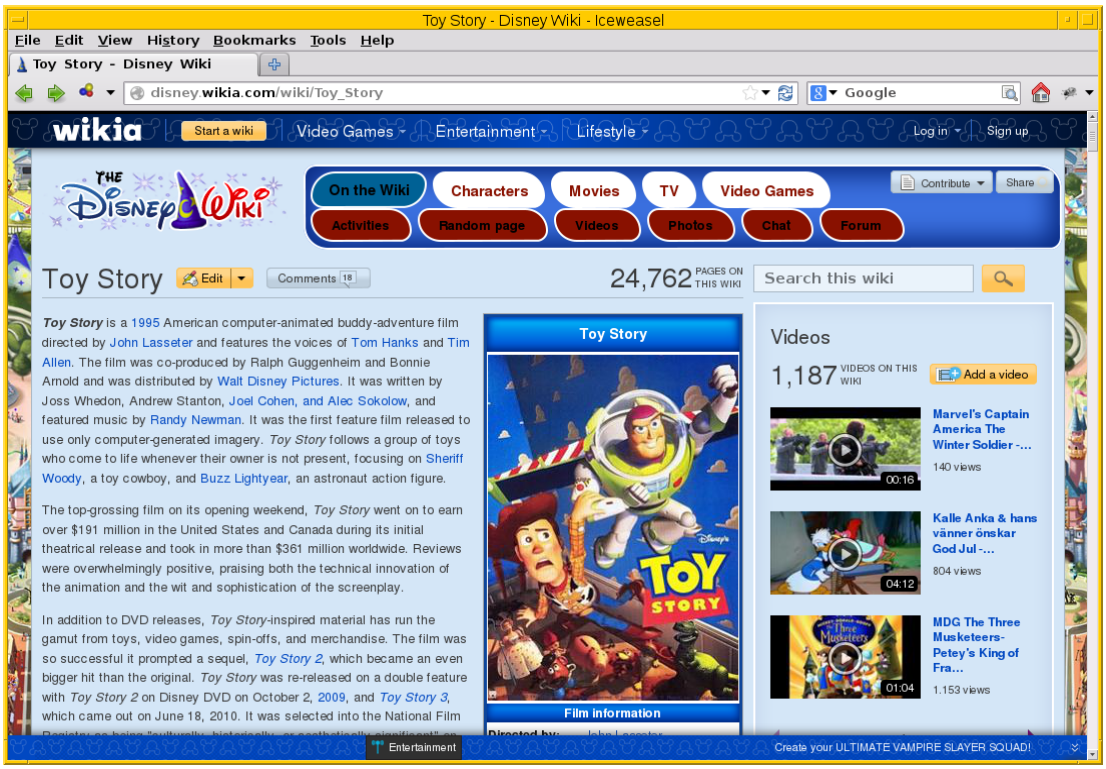

Abbildung 2.8: Beschreibung der Filmserie *Toy Story* im Disney Wiki

Auch bei der Bezeichnung der Aktualisierungen zur stabilen Veröffentlichung ergeben sich über die Jahre hinweg kleine Unterschiede. Anfangs erfolgte die Kennzeichnung durch Anhängen des Buchstabens `r` und der Nummer der Aktualisierung, z.B. `4.0r8` für die 8. Aktualisierung von Debian 4.0 *Etch*. Seit Debian 5.0 *Lenny* wird stattdessen ein Punkt verwendet, so z.B. `5.0.3` für die dritte Aktualisierung.

Seit Debian 4.0 *Etch* bekamen stabile Veröffentlichungen immer eine neue Nummer an erster Stelle. Seit Debian 7 *Wheezy* ist die Null an zweiter Stelle verschwunden und stattdessen wird die Nummer der Aktualisierung genutzt, so z.B. `7.3` für die dritte Aktualisierung von Debian 7 *Wheezy*.

2.10.3 Zusammenhang von Alias-Namen und Entwicklungsständen

Neben den o.g. Entwicklungsständen haben alle Veröffentlichungen auch noch Alias-Namen, die eine Veröffentlichung stets unverändert beibehält. Jede neue Veröffentlichung startet nach einer stabilen Veröffentlichung als *testing*, wird dann bei der nächsten stabilen Veröffentlichung zu *stable*, bei der übernächsten zum *oldstable* und danach zu *oldoldstable*.

Ist eine Veröffentlichung — sei es als *oldstable* oder als *oldoldstable* — am Ende ihrer Unterstützung angelangt, wird sie in das Debian-Archiv [Debian-Archive] übertragen. Dieses Archiv beinhaltet alle nicht mehr unterstützten Veröffentlichungen.

Eine weitere Ausnahme bildet die Veröffentlichung zu *unstable*. Sie besitzt stets den gleichen Alias-Namen *Sid*. In der Filmreihe Toy Story ist das passenderweise der Name des bösen Nachbarkinds, welches immer alle Spielzeuge kaputt macht. *Sid* ist auch gleichzeitig ein Akronym für *still in development* – zu deutsch „noch in Entwicklung“ –, was den Status der Veröffentlichung der zukünftigen Distribution sehr treffend umschreibt.

Experimental trägt – analog zu *unstable* – immer den Alias-Namen *rc-buggy*, was im Debian-Jargon eine Kurzform für „contains release-critcal bugs“ darstellt. Das läßt sich sinngemäß als „in dieser Form ungeeignet zur Aufnahme in eine Veröffentlichung“ übersetzen.

2.10.4 Pakete auf Wanderschaft von einem Entwicklungsstand in den nächsten

Sieht man von Uploads nach *experimental* ab, fängt das Leben einer neuen Version eines Debianpakets mit dem Hochladen nach *unstable* an. Das Paket wird automatisch in *testing* übernommen, sobald einige Bedingungen erfüllt sind:

- Die Version des Pakets in *unstable* führt keine neuen veröffentlichungskritischen Fehler in *testing* ein.
- Alle notwendigen Abhängigkeiten des Pakets sind in *testing* verfügbar oder werden gleichzeitig nach *testing* migriert.
- Es darf keine Abhängigkeiten von Paketen zerstören, die bereits in *testing* enthalten sind und damit deren Installation verhindern.
- Das Paket hat ein Mindestalter an Tagen erreicht. Dieses Mindestalter hängt vom Wert des Felds *urgency* (engl. für Dringlichkeit) im aktuellen Changelog-Eintrag ab und beträgt entweder 10, 5 oder 2 Tage. Die Dringlichkeit wird dabei vom Paketmaintainer entschieden. Bei Korrekturen von sicherheitsrelevanten Fehlern ist es durchaus üblich, dass die Dringlichkeit auf „hoch“ gesetzt wird und damit nur 2 Tage beträgt.
- Das Paket muss auf allen Architekturen, auf denen es gebaut wird, in der aktuellsten Version verfügbar sein.
- Das Paket muss auf allen Architekturen bereitstehen, auf denen es vorher bereits gebaut wurde. Für Ausnahmen muss zuerst das alte Paket aus dem Archiv manuell entfernt werden.

Das Debian-Release-Team kann allerdings diese Bedingungen individuell übersteuern und kürzere oder längere Fristen für den Übergang in die *testing*-Veröffentlichung setzen.

Zu einem bestimmten Zeitpunkt im Entwicklungszyklus einer neuen stabilen Veröffentlichung friert das Release-Team die *testing*-Veröffentlichung ein – auch genannt *Freeze* (engl. für Einfrieren). Ab diesem Moment wandern keine Pakete mehr automatisch von *unstable* nach *testing* und das Debian-Release-Team muss jeden einzelnen, weiteren Übergang eines Pakets explizit abnicken. Je länger der Freeze andauert, desto schärfer werden die Bedingungen, unter denen das Debian-Release-Team einen Übergang nach *testing* akzeptiert. Im Normalfall werden nur noch Paketversionen akzeptiert, die ausschließlich Fehler korrigieren und keine neuen Features einführen. Daher wird für diesen Zustand auch der Begriff *Feature Freeze* verwendet.

Auf diese Weise wird versucht, sämtliche veröffentlichungskritischen Fehler in der *testing*-Veröffentlichung zu beheben. Sobald es dort keinen dieser Fehler mehr gibt, geschehen die folgenden Dinge:

- Die bisherige Veröffentlichung *stable* wird zu *oldstable*. Sie behält dabei ihren Alias-Namen bei.
- Eine Kopie des aktuellen Zweigs *testing* wird zum neuen Zweig *stable*. Der Alias-Name zieht mit um.
- *testing* bekommt einen neuen Alias-Namen.
- Der Freeze wird aufgehoben und die Pakete propagieren wieder automatisch von *unstable* nach *testing*.

2.10.5 Organisation der Pakete im Paketpool

Wenn eine Paketversion von *unstable* nach *testing* wandert oder aus *testing* das neue *stable* wird, werden allerdings nicht wirklich Pakete kopiert. Stattdessen werden vielmehr nur die Metadaten des betreffenden Pakets von einer Paketliste in eine andere umgetragen. Das Paket selbst liegt immer noch an gleicher Stelle und nur einmal im sogenannten Paketpool.

So kann es vorkommen, dass ein Paket, welches nur selten aktualisiert wird, in allen aktuellen Veröffentlichungen in der gleichen Version vorkommt und dafür auch nur einmal auf jedem Spiegel des Debian-APT-Archivs liegt. Welches Paket dann aus den verschiedenen Entwicklungsständen bei einer Installation ausgewählt wird, erfahren Sie unter „Aus welchem Repo kommen die Pakete“ (siehe Abschnitt 8.18) genauer.

2.11 Benennung einer Paketdatei

Während der Benutzung von `dpkg`, APT oder `aptitude` sind Sie sicher schon mit den etwas langen und auf den ersten Blick kurios anmutenden Dateinamen der einzelnen Pakete in Berührung gekommen. Die Benennung einer Paketdatei folgt einem ausgeklügelten Schema [Krafft-Debian-System144]. Dieses Schema ist eine Konvention, die leider bei Paketen aus Drittquellen oft nicht eingehalten wird.

Der Dateinamen besteht aus den drei Felder *Paketname*, *Version* und *Architektur*, welche durch einen Unterstrich _ voneinander abgegrenzt werden, plus einem Punkt und der Dateiendung *.deb*. Gemäß den Debian-Richtlinien [Debian-Policy-Manual] sind in o.g. Feldern nur ASCII-Zeichen zulässig. Unterstriche, Leerzeichen und Umlaute sind nicht gestattet. Das hilft dabei, Missverständnissen vorzubeugen und die Paketenamen mit allen Zeichensätzen anzeigen zu können.

2.11.1 Paketname

Feld 1 bezeichnet den Namen des Pakets, welche durch die Paketdatei bereitgestellt wird. Die Paketdatei `iceweasel_3.5.16-12_i386.deb` ist ein Binärpaket (Dateiendung `.deb`) und beinhaltet den Webbrowser Iceweasel.

Darüberhinaus existieren bei der Benennung eine Reihe von Gepflogenheiten in Form von Präfixen und Suffixen. Diese stellen kein „muss“ dar, vereinfachen aber die Handhabung insgesamt sowie die Paketklassifikation und die spätere Recherche nach Paketen.

Beginnt der Paketname mit der Zeichenkette `lib`, handelt es sich meist um eine Bibliothek, auf englisch *library*. Als Beispiel seien hier die beiden XML-Bibliotheken *libxml2-utils* und *libxml2* genannt. Aus dem Schema fallen allerdings die Komponenten zu LibreOffice wie *libreoffice-writer* (Textverarbeitung *Writer*) und *libreoffice-calc* (Tabellenkalkulation *Calc*) heraus.

Endet der Paketname mit dem Suffix `-dev` wie bspw. in *libxslt1-dev*, beinhaltet das Paket Kopfdateien (engl. *header files*), die nur notwendig sind, wenn Sie Programme unter Nutzung der dazugehörigen Bibliothek entwickeln. *dev* ist die gebräuchliche englische Abkürzung für *development*. Im vorgenannten Beispiel befinden sich in dem Paket die Kopfdateien zur XSLT-1-Bibliothek.

Das Suffix `-doc` weist auf Dokumentation hin, welches häufig noch von einer Abkürzung für die jeweilige Sprache gefolgt wird. Der Paketname *aptitude-doc-es* beinhaltet bspw. die spanische Übersetzung der Dokumentation zu `aptitude`.

Die Suffixe `-common` und `-data` deuten an, dass das Paket Dateien beinhaltet, die von mehreren Teilen eines Programms gemeinsam genutzt werden. Als Beispiel sei hier *wireshark-common* genannt, welches sowohl die Daten für die graphische Variante des Netzwerktools *wireshark*, als auch für die textbasierte Version *tshark* beinhaltet.

2.11.2 Versionsnummer

Feld 2 spiegelt eine Reihe unterschiedlicher Informationen und Zustände wieder, aus dem Sie den Versionsstand und -verlauf eines Pakets erkennen. Die Versionsangabe kann sowohl numerische Zeichen (Ziffern), als auch nichtnumerische Zeichen wie Punkte, Tilden und Buchstaben beinhalten.

Handelt es sich um ein *nicht-natives Debian-Paket*, besteht die Versionsnummer aus der Upstream-Version und der Debian-Revision. Bei dem Paket *smartpm_1.4-2_all.deb* für *smartpm* (siehe Abschnitt 6.4.3) ist die Angabe 1.4 die Upstream-Version und die darauffolgende mit einem Minus - abgetrennte `2` steht für die zweite Debian-Revision. Hier liegt also das zweite Debianpaket vor, welches auf der Upstream-Version 1.4 basiert. Beinhaltet die Versionsnummer mehrere Bindestriche, ist immer der letzte Bindestrich der Trenner zwischen der Upstream-Version und der Debian-Revisionsnummer.

Handelt es sich hingegen um ein *natives Debian-Paket*, d.h. eine Software, die ausschließlich als Debian-Paket vertrieben wird, gibt es keine Debian-Revisionsnummer und die Versionsnummer des Pakets ist identisch mit der Versionsnummer der Software. Für das Paket *dpkg_1.17.25_i386.deb* zu `dpkg` ist das 1.17.25.

Ändert sich bei der Aktualisierung (Upstream) die Versionsangabe so grundlegend, dass die neuere Version eine kleinere Versionsnummer hat als die vorherige Version, so muss der Paketversion die Angabe einer mit einem Doppelpunkt abgetrennten *Epoche* hinzugefügt werden. Ist bspw. die vorhergehende Versionsnummer 2013.06.06-4 (Upstream-Version 2013.06.06 Revision 4), entspricht das der Epoche 0 und ist identisch zu 0:2013.06.06-4. Die Folgeversion wird dann 1:1.0-1, d.h. Epoche 1, Upstream-Version 1.0 und Revision 1.

Um eine spätere *alphanumerisch korrekte Sortierung anhand des Releasestatus* zu ermöglichen, sind eine bzw. mehrere aufeinanderfolgende Tilden ~ zulässig. Damit wird bspw. die Version `1.0~beta1` vor der Version `1.0` einsortiert. Diese Schreibweise kam zuerst bei Debian auf, wurde mittlerweile aber auch von anderen Open-Source-Projekten übernommen.

Zudem sind eine Reihe von *Suffixen* gebräuchlich. Diese gelten zwar nur als Konvention, werden aber auch an einigen Stellen erwartet.

+nmu<n>
: Non-Maintainer-Upload (NMU) eines nativen Pakets. Das bezeichnet eine Paketversion, die nicht vom Verantwortlichen (Maintainer) des Pakets stammt. Bspw. bezeichnet die Datei `adduser_3.113+nmu3_all.deb` das Paket *adduser* als dritten Non-Maintainer-Upload basierend auf der Version 3.113 des Maintainers.

-<x>.<y>
: Debian-Revisionsnummer eines Non-Maintainer-Upload (NMU) eines nicht-nativen Pakets. Dabei bezeichnet `<x>` die letzte Revision des Maintainers (oder `0`, falls es keine solche gab) und `<y>` die Nummer des NMU basierend auf dieser Revision des Maintainers. So ist z.B. die Datei `bash_4.2+dfsg-0.1_i386.deb` das Debianpaket *bash* als Non-Maintainer-Upload einer neuen Upstreamversion basierend auf der Veröffentlichung 4.2. Hingegen bezeichnet die Angabe 4.2-2.1 den ersten Non-Maintainer-Upload, welcher auf der Basis der Maintainer-Version 4.2-2 erstellt wurde.

+b<n>
: Kennzeichnung eines Binären Non-Maintainer-Uploads (*BinNMU*). Das bezeichnet eine Übersetzung des Pakets ohne vorherige Änderung des Quellcodes. Das tritt bspw. dann auf, wenn sich die Abhängigkeiten zum Bauen des Pakets geändert haben (sogenannte *build-dependencies*). Die Angabe `123-4+b2` steht dabei für den zweiten Erstellungsdurchlauf des Pakets aus den Quellen der Version `123-4`. Ubuntu verwendet dafür stattdessen die Syntax `123-4build2`.

~bpo<x>+<y>
: Backports (siehe Kapitel 19) bezeichnen eine Rückportierung einer neueren Version auf die aktuelle Veröffentlichung. Dabei steht das Kürzel `bpo` für `backports.org`, dem Namen des Backports-Projektes, bevor es in Debian integriert wurde. Die Angabe `123-3~bpo8+2` steht bspw. für eine Rückportierung der Upstream-Version 123-3 auf Debian 8 *Jessie*. Die Ziffer 2 deklariert das Paket die zweite Backports-Revision des Paket.

+deb<x>u<y>
: stabiles Update. Die Angabe `121-3+deb7u2` steht für das zweite stabile Update des Pakets mit der Version 121-3 in Debian 7 *Wheezy* (<x>=7 und <y>=2).

ubuntu<n>
: ein Debianpaket, welches für Ubuntu angepaßt wurde. `<n>` bezeichnet die Ubuntu-Revisionsnummer, so bspw. `121-3ubuntu4` für die vierte Ubuntu-Revision des Debian-Pakets mit der Versionsnummer `121-3`.

2.11.3 Architektur oder Plattform

Feld 3 in der Versionsangabe gibt an, für welche Architektur das vorliegende Paket übersetzt wurde. Die Benennung entspricht den Bezeichnungen, wie sie unter Debian-Architekturen in Abschnitt 1.2 aufgelistet sind. Die Angabe `asterisk_1.8.13.1~dfsg-3+deb7u1_armhf.deb` beschreibt die Paketierung der Telefoniesoftware Asterisk für die ARM-Plattform mit Hardware-Floating-Point-Unterstützung. Im Gegensatz dazu ist das Paket `asciidoc_8.5.2-1_all.deb` plattformunabhängig einsetzbar.

2.12 Multiarch einsetzen

`dpkg` führt eine Liste mit allen Architekturen, für die es Pakete installiert bzw. installieren darf. Diese Liste befindet sich in der Datei `/var/lib/dpkg/arch` und existiert allerdings nur, sofern Sie zuvor auch Fremdarchitekturen ergänzt haben.

Inhalt der Liste der Architekturen eines Systems mit amd64 als Basisarchitektur und i386 als Fremdarchitektur

```
$ cat /var/lib/dpkg/arch
amd64
i386
$
```

Die erste Architektur in dieser Datei ist die Basisarchitektur. Diese geben Sie mit der `dpkg`-Option `--print-architecture` aus. Früher bzw. bei älteren `dpkg`-Versionen heißt die Option `--print-installation-architecture`. Die Fremdarchitekturen zeigen Sie mit `dpkg --print-foreign-architectures` an.

Über die beiden `dpkg`-Optionen `--add-architecture` und `--remove-architecture` erweitern bzw. reduzieren Sie die Liste entsprechend. Beim Aufruf geben Sie dazu jeweils noch die gewünschte Architektur als Parameter an, bspw. `dpkg --add-architecture i386`, wenn Sie zusätzlich die Architektur für 32-Bit-PCs nutzen wollen, weil es die von Ihnen gewünschte Software nur für 32-Bit-Systeme gibt.

Während des Vorgangs schreibt `dpkg` diese Änderung zuerst in eine temporäre Datei namens `/var/lib/dpkg/arch-new`. Wurden alle anderen Änderungen erfolgreich vorgenommen, benennt `dpkg` diese Datei in `/var/lib/dpkg/arch` um.

Installation von Paketen für fremde Architekturen
Bitte berücksichtigen Sie bei Ihrer Softwareplanung, dass nicht jedes Paket für alle Plattformen verfügbar ist. Wenn es verfügbar ist und Sie es erfolgreich auf Ihrem System installieren konnten, heißt das nicht automatisch, dass es auch auf Ihrer Architektur funktioniert, sondern nur, dass die Paketverwaltung alle benannten Paketabhängigkeiten erfüllen konnte.

Löschen einer Fremdarchitektur
Das Entfernen einer Fremdarchitektur gelingt Ihnen nur dann, wenn keine Pakete (mehr) für diese Architektur auf Ihrem System installiert sind. Wie Sie Pakete architekturbezogen deinstallieren, lesen Sie in Abschnitt 8.41 nach.

2.12.1 Multiarch-Beispiel: Installieren eines 32-Bit-Pakets auf einem 64-Bit-System

Ein *vollständiges Beispiel* für den Einsatz von *multiarch* ist die Nutzung des Forth-Interpreters `pforth` auf einem 64-bittigen Debian (Architektur *amd64*). `pforth` ist über das gleichnamige Paket bislang nur nativ für 32-Bit-Betriebssysteme verfügbar. Gleiches betrifft das recht weit verbreitete, aber nicht-quelloffene Kommunikationsprogramm Skype [Skype]. Eine passende Schritt-für-Schritt-Anleitung finden Sie im Debian Wiki [Debian-Wiki-Skype].

Im Folgenden zeigen wir Ihnen anhand des vorgenannten Pakets `pforth`, wie eine solche Installation abläuft und insbesondere, welche Einzelschritte wir dabei für beachtenswert halten. Zunächst überprüfen Sie mittels `dpkg` und dessen Option `--print-architecture` die derzeit benutzte Architektur Ihres Systems – im hier betrachteten Fall ist es *amd64*. Danach ergänzen Sie die Liste der Architekturen via `dpkg --add-architecture i386` um *i386* als weitere Plattform, für die Ihr System Pakete akzeptiert. Ob der Vorgang erfolgreich war, zeigt Ihnen der Parameter `--print-foreign-architectures` von `dpkg` an. Damit erhalten Sie eine Übersicht zu allen „Fremdarchitekturen“, die ihr Debiansystem derzeit akzeptiert.

Hinzufügen einer weiteren genutzten Paketarchitektur mittels dpkg

```
# dpkg --print-architecture
amd64
# dpkg --add-architecture i386
# dpkg --print-foreign-architectures
i386
#
```

Nun aktualisieren Sie die lokale Liste der verfügbaren Pakete mittels `apt-get update`, wobei APT nun auch die Informationen zu den Paketen der neu hinzugefügten Architektur herunterlädt.

Aktualisieren der Paketlisten

```
# apt-get update
Ign http://ftp.ch.debian.org jessie InRelease
Hit http://ftp.ch.debian.org jessie Release.gpg
Hit http://ftp.ch.debian.org jessie Release
Hit http://ftp.ch.debian.org jessie/main amd64 Packages
Get:1 http://ftp.ch.debian.org jessie/main i386 Packages [6769 kB]
Hit http://ftp.ch.debian.org jessie/main Translation-en
Fetched 6769 kB in 6s (1005 kB/s)
Reading package lists... Done
```

Als nächsten Schritt prüfen Sie mit dem Aufruf `apt-cache policy`, für welche akzeptierte Architektur das von Ihnen gewünschte Paket bereitsteht. Die Details zum Aufruf von `apt-cache` finden Sie in Abschnitt 8.18.

Prüfung auf Verfügbarkeit für eine Architektur mittels apt-cache

```
# apt-cache policy pforth
pforth:i386:
  Installed: (none)
  Candidate: 21-12
  Version table:
     21-12 0
        990 http://ftp.ch.debian.org/debian/ jessie/main i386 Packages
#
```

Sie ersehen aus der obigen Ausgabe, dass das Paket bislang noch nicht auf Ihrem System installiert ist. Es steht für die Architektur *i386* und die Veröffentlichung Debian 8 *Jessie* bereit. Nun können Sie das Paket *pforth* installieren. Das zieht u.a. das essentielle Paket *libc6* für die Architektur *i386* nach sich, um die Abhängigkeiten zum Paket *pforth* zu erfüllen.

Installation des i386-Pakets pforth auf amd64-Debian

```
# apt-get install pforth
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  gcc-4.9-base:i386 libc6:i386 libgcc1:i386
Suggested packages:
  glibc-doc:i386
Recommended packages:
  libc6-i686:i386
The following NEW packages will be installed:
  gcc-4.9-base:i386 libc6:i386 libgcc1:i386 pforth:i386
0 upgraded, 4 newly installed, 0 to remove and 27 not upgraded.
Need to get 4,252 kB of archives.
After this operation, 9,727 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.ch.debian.org/debian/ jessie/main gcc-4.9-base i386 4.9.1-15 [158 kB]
Get:2 http://ftp.ch.debian.org/debian/ jessie/main libc6 i386 2.19-11 [3,977 kB]
Get:3 http://ftp.ch.debian.org/debian/ jessie/main libgcc1 i386 1:4.9.1-15 [48.2 kB]
Get:4 http://ftp.ch.debian.org/debian/ jessie/main pforth i386 21-12 [69.1 kB]
Fetched 4,252 kB in 0s (20.5 MB/s)
Preconfiguring packages ...
Selecting previously unselected package gcc-4.9-base:i386.
(Reading database ... 474485 files and directories currently installed.)
Preparing to unpack .../gcc-4.9-base_4.9.1-15_i386.deb ...
Unpacking gcc-4.9-base:i386 (4.9.1-15) ...
Selecting previously unselected package libc6:i386.
Preparing to unpack .../libc6_2.19-11_i386.deb ...
Unpacking libc6:i386 (2.19-11) ...
Replacing files in old package libc6-i386 (2.19-11) ...
Selecting previously unselected package libgcc1:i386.
Preparing to unpack .../libgcc1_1%3a4.9.1-15_i386.deb ...
Unpacking libgcc1:i386 (1:4.9.1-15) ...
Selecting previously unselected package pforth.
Preparing to unpack .../archives/pforth_21-12_i386.deb ...
Unpacking pforth (21-12) ...
Processing triggers for man-db (2.7.0-1) ...
Setting up gcc-4.9-base:i386 (4.9.1-15) ...
Setting up libc6:i386 (2.19-11) ...
Setting up libgcc1:i386 (1:4.9.1-15) ...
Setting up pforth (21-12) ...
Processing triggers for libc-bin (2.19-11) ...
#
```

In o.g. Fall wurde das Paket `libc6` als Abhängigkeit auch für die Architektur *i386* installiert. Sie erkennen das daran, dass neben dem Namen des Pakets auch die Architektur angegeben wird. Als Trennzeichen in der Ausgabe fungiert hier ein Doppelpunkt.

Abschließend überprüfen Sie mittels `dpkg`, für welche Architekturen die Pakete `pforth` und `libc6` auf Ihrem System installiert sind.

Installationsstatus für das Paket libc6

```
# dpkg -l pforth libc6
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version      Architecture Description
+++-==============-============-============-=================================
ii  libc6:amd64    2.19-11      amd64        GNU C Library: Shared libraries
ii  libc6:i386     2.19-11      i386         GNU C Library: Shared libraries
ii  pforth         21-12        i386         portable Forth interpreter
#
```

Im letzten Schritt probieren Sie aus, ob das frisch installierte 32-Bit-Programm auch unter Ihrem 64-Bit-Betriebssystem funktioniert. Dazu rufen Sie das Programm auf.

Ausführung von pforth

```
$ pforth
PForth V21
pForth loading dictionary from file /usr/lib/pforth/pforth.dic
     File format version is 8
     Name space size = 120000
     Code space size = 300000
     Entry Point     = 0
     Little  Endian Dictionary
Begin AUTO.INIT ------
...
$
```

2.13 Paket-Priorität und essentielle Pakete

Jedes Paket beinhaltet ein Feld namens `Priority` – englisch für „Priorität“. Dabei geht es aber weniger um eine Rangfolge von Paketen, sondern um die Wichtigkeit eines Pakets bzw. um die Wahrscheinlichkeit, dass Sie dieses Paket installieren möchten.

Debian kennt die folgenden fünf Prioritätsstufen:

- erforderlich (*required*)
- wichtig (*important*)
- standard (*standard*)
- optional (*optional*)
- extra (*extra*)

Die Begriffe in Klammern geben die Schlüsselworte wieder, die in der Paketbeschreibung genutzt werden. Jede dieser o.g. Stufen hat eine bestimmte Bedeutung.

Auflistung der Pakete mit einer festgelegten Priorität
In Abschnitt 8.9 lesen Sie, wie Sie mit `aptitude` Pakete mit einer spezifischen Priorität finden.

2.13.1 Prioritätsstufe „erforderlich“ (*required*)

Dieser Prioritätsstufe sind Pakete zugeordnet, die für die korrekte Funktion des Betriebssystem unbedingt erforderlich sind. Dazu gehören beispielsweise *dpkg*, *coreutils* für die GNU Core Utilities mit den Befehlen wie `ls`, `rm`, `cp`, `mv`, das Init-System (seit Debian 8 Jessie das Metapaket *init*) und die C-Standard-Bibliotheken (*libc6* auf den meisten Architekturen).

Entfernen Sie eines oder mehrere Pakete mit dieser Prioritätsstufe, kann das Ihre Installation so stark beschädigen, dass selbst das Werkzeug `dpkg` nicht mehr funktioniert.

Systeme, die nur aus Paketen der Prioritätsstufe „erforderlich“ bestehen, sind zwar lauffähig, aber im Normalfall nahezu unbenutzbar, da z.B. Pakete wie APT, *less* oder ein Texteditor fehlen. Die letztgenannten sind zum Betrieb nicht zwingend erforderlich[2].

2.13.2 Prioritätsstufe „wichtig“ (*important*)

In diese Prioritätsstufe gehören alle Pakete, die auf jedem UNIX- bzw. Debian-System zu erwarten sind oder ohne die das System nur sehr schwierig zu warten wäre. Das schließt auch Server ohne Monitor mit ein.

Als Pakete gehören neben *apt* u.a. *gnupg* und der Debian-Archiv-Schlüsselring zum Überprüfen der Signaturen von Paketlisten (siehe Abschnitt 8.35) dazu, ebenso OpenSSL, ein DHCP-Client, zwei Texteditoren (eine abgespeckte Variante von Vim sowie Nano), Kommandozeilenwerkzeuge zur Prozessverwaltung (`ps`, `kill`, `free`, `top`, `uptime` aus dem Paket *procps*), ein Syslog-Daemon, ein Cron-Daemon, Man-Pages, Netzwerk-Programme wie `ping`, `traceroute` und `iptables` sowie das Netzwerkschnittstellenverwaltungssystem `ifupdown`.

Diese Prioritätsstufe beinhaltet weder große Applikationen noch graphische Programme. Insbesondere gehören weder GNU Emacs noch TeX noch das X Window System oder das *xterm* in diese Kategorie.

2.13.3 Prioritätsstufe „standard“ (*standard*)

Haben Sie alle Pakete dieser Prioritätsstufe installiert, verfügen Sie über ein nicht allzu großes, aber auch nicht zu unkomfortables System ohne graphische Oberfläche. Ein solches System wird im Debian Installer ausgewählt, wenn Sie als Administrator bei der Installation nicht explizit etwas anderes festlegen. Es enthält nur wenige größere Anwendungen und Daemons.

Dazu gehören u.a. ein abgespeckter Exim als lokales Mail-Server-Programm, die E-Mail-Programme `mutt` und `mailx`, eine vollständige Perl-Installation (d.h. Perl mitsamt allen „Core“-Modulen[3]), Python, Client-Anwendungen für SSH, FTP, Telnet, NFS und Whois, ein Text-Modus-Webbrowser (`w3m`) und der allgegenwärtige Textdateien-Betrachter `less`. Außerdem ist `reportbug` enthalten, ein Programm zum Melden von Fehlern in Debian (siehe dazu Abschnitt 32.3).

2.13.4 Prioritätsstufe „optional“ (*optional*)

Dies ist in gewisser Weise der Standardwert für die Priorisierung eines Pakets. Alle Pakete, die in keine der anderen Stufen gehören, werden dieser Prioritätsstufe zugeordnet. Sie enthält deswegen auch den Großteil aller Pakete in Debian. Optional bedeutet in diesem Kontext, dass diese Pakete nicht von jedermann benötigt werden.

2.13.5 Prioritätsstufe „extra“ (*extra*)

In dieser Prioritätsstufe sind einerseits Pakete, die im Konflikt mit Paketen aus den anderen Prioritätsstufen stehen. Dazu zählen z.B. alternative Mail-Transport-Agents wie Postfix, alternative Cron-Daemons wie Cronie oder alternative Syslog-Daemons wie Syslog-NG oder die Syslog-Implementation aus Busybox.

Andererseits enthält sie aber auch Pakete, die nur in ganz bestimmten Fällen gebraucht werden, z.B. Programme zur Nutzung exotischer Hardware oder nur in bestimmten Umfeldern vorkommenden Daten, Pakete mit Debug-Symbolen für andere Pakete, Übergangspakete, etc. Beispielsweise sind viele Pakete aus dem Bereich „Wissenschaft“ mit dieser Priorisierung versehen.

[2] Hat z.B. ein System keine Netzwerkanbindung und wird deswegen nur sehr selten aktualisiert, ist APT nicht notwendig. Aktualisierungen können auch auf anderen Wegen, bspw. via USB-Stick oder SD-Karte mittels `dpkg` eingepflegt werden. Allerdings sind dann Abhängigkeiten ggf. manuell aufzulösen. Bei reinen Paketaktualisierungen ist dies nur sehr selten ein Problem, da die Abhängigkeiten im Normalfall auch schon von der vorherigen Paketversion gebraucht wurden.

[3] Perl selbst und ein paar wenige Perl-Module sind im Paket *perl-base* welches „essentiell“ ist.

2.13.6 Markierung „essentiell“ (*essential*)

Zusätzlich zu den bereits oben vorgestellten Prioritäten gibt es noch die Markierung *essential*. Diese Markierung tragen nur sehr grundlegende Pakete.

Eintrag in der Paketbeschreibung

```
Essential: yes
```

Pakete mit dieser Markierung müssen nicht explizit als Abhängigkeit bei anderen Paketen deklariert werden. In der Regel sind alle Pakete der Prioritätsstufe „erforderlich“ in dieser Form markiert, von denen kein anderes Paket dieser Stufe abhängt. Somit wird auch bei der Entfernung eines nicht-essentiellen Pakets der Stufe „erforderlich“ gewarnt. Das passiert jedoch nicht, wenn bei einer Umbenennung eines solchen Pakets das alte Paket entfernt wird, um für das neue Paket Platz zu machen oder weil es nicht mehr gebraucht wird (d.h. irgendwann nicht mehr notwendig ist).

In Abschnitt 8.9 lesen Sie, wie Sie auflisten, welche Pakete genau auf Ihrer Version von Debian als essentiell markiert sind.

Diese Markierung hat weiter den Effekt, dass sich beispielsweise auch `dpkg` weigert, solche Pakete zu entfernen (siehe Abschnitt 8.41). Mit dem zusätzlichen Parameter `--force-remove-essential` übergehen Sie diese Voreinstellung und können die Aktion trotzdem durchführen (siehe dazu „Paketoperationen erzwingen“ in Abschnitt 8.43).

`apt-get` und `aptitude` entfernen diese Pakete nur nach Eingabe des vollständigen Satzes „Ja, tue was ich sage!“ (`apt-get`) bzw. „Mir ist klar, dass das eine sehr schlechte Idee ist.“ (`aptitude`). Diese Sätze werden jeweils in der eingestellten Sprache Ihres Debiansystems angezeigt (Lokalisierung), sofern eine Übersetzung vorhanden ist. Nachfolgendes Beispiel zeigt die Bildschirmausgabe vor der Entfernung des Pakets *init* [Debian-Paket-init].

Eingabeaufforderung von apt-get vor der Entfernung des essentiellen Pakets init

```
# apt-get remove init
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden Pakete werden ENTFERNT:
  init
WARNUNG: Die folgenden essentiellen Pakete werden entfernt.
Dies sollte NICHT geschehen, außer Sie wissen genau, was Sie tun!
  init
0 aktualisiert, 0 neu installiert, 1 zu entfernen und 0 nicht aktualisiert.
Nach dieser Operation werden 29,7 kB Plattenplatz freigegeben.
Sie sind im Begriff, etwas potentiell Schädliches zu tun.
Zum Fortfahren geben Sie bitte »Ja, tue was ich sage!« ein.
 ?]
```

2.14 Verbreitungsgrad von Paketen

Wie bereits deutlich wurde, besteht die Distribution Debian GNU/Linux aus einer sehr großen Anzahl Paketen. In dieser Vielfalt spiegeln sich die Interessen der Benutzer sehr deutlich wieder.

Das *Debian Quality Assurance Team* (kurz QA Team) [DebianQA] sorgt dafür, dass eine möglichst hohe Softwarequalität in Debian gehalten wird. Neben den Werkzeugen zur Qualitätssicherung (siehe Kapitel 32) gehören dazu die Trendforschung, die Bestandsaufnahme und eine Auswertung darüber, ob und vor allem wie häufig ein Paket installiert wird. Das sagt zwar nicht unbedingt etwas darüber aus, ob es tatsächlich verwendet wird, aber es zeigt, ob an einem Softwarepaket prinzipiell Interesse besteht. Dieser Aspekt fließt mit ein, um zu entscheiden, ob ein Paket weiterhin Bestandteil des Softwareumfangs von Debian bleibt.

Diese Analyse geht direkt auf den Ursprung von Debian zurück und versucht eine Antwort darauf zu geben, welche Software die Benutzer verwenden. Unmittelbare Ergebnisse sind die Auswahl der Softwarepakete, die in Debian bereitstehen und für diese Distribution gepflegt werden, weiterhin die Einordnung in die entsprechenden Kategorien (siehe Abschnitt 2.8) und die Priorisierung (siehe Abschnitt 2.13). Für die Zusammenstellung von Installationsimages spielt der Nutzungsgrad eine große Rolle – Pakete, die häufiger genutzt werden, haben eine größere Chance, auf die ersten Installationsimages zu gelangen.

Grundlage für die erfassten Daten ist das Projekt Popcon – der *Debian Popularity Contest* [Debian-Popularity-Contest]. Die Benutzung ist freiwillig und über dessen Teilnahme entscheiden Sie als Benutzer selbst. Voraussetzung dafür ist die Installation des Pakets *popularity-contest* und dessen Aktivierung.

Danach wird in regelmäßigen Abständen (wöchentlich) der Softwarebestand (d.h. die installierten Pakete) erfasst, an das Popcon-Projekt übertragen und danach anonymisiert ausgewertet. Über die Projektwebseite erfolgt eine tabellarische Übersicht und eine graphische Auswertung. Abbildung 2.9 zeigt beispielhaft das Ergebnis für das Paket *nginx*.

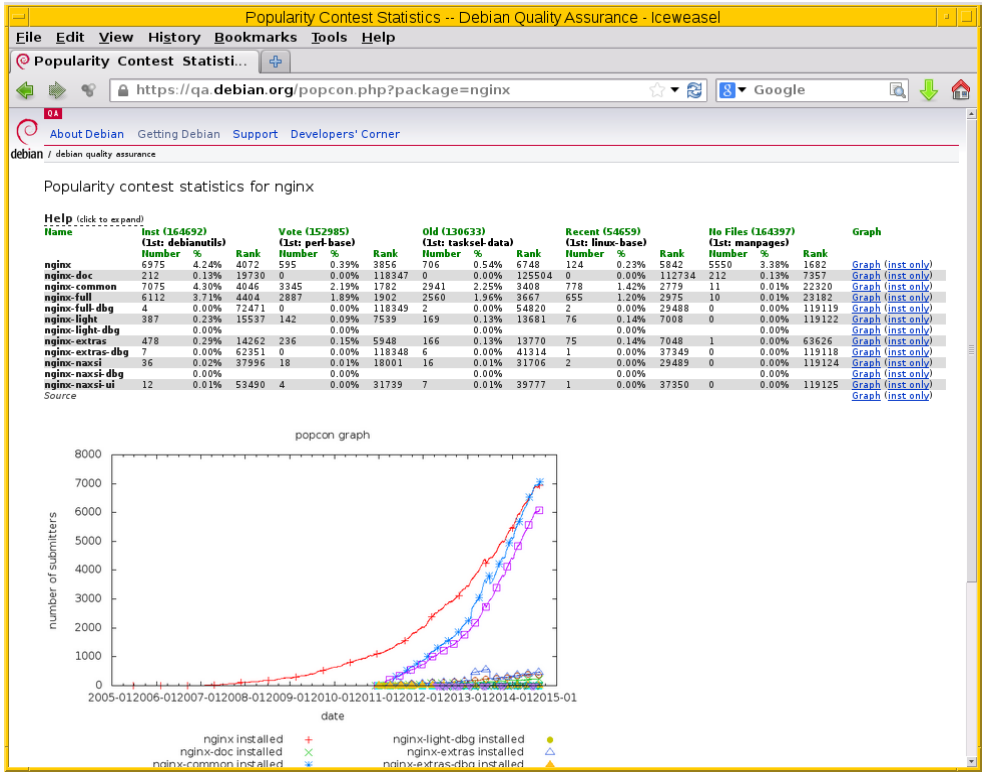

Abbildung 2.9: Erfasster Verbreitungsgrad für das Paket *nginx*

2.14.1 Verschiedene Metriken

Neben der Architektur der Installation und welche Pakete installiert sind, erfasst Popcon anhand der Zeitstempel im Dateisystem ausserdem noch folgende Daten für jedes installierte Paket:

- Wann wurde das Paket zuletzt aktualisiert oder installiert? Dies wird für den Graphen *recent* (kürzlich) verwendet und anhand des Zeitstempels der Dateien des Pakets unter `/var/lib/dpkg/info` eruiert.
- Wann wurde zuletzt auf ausführbare Dateien des Pakets zugegeriffen? Dies wird für die Graphen *vote* (dafür stimmen) *old* (alt) verwendet und anhand der Zugriffszeitstempel (*atime*) von Programmdateien des Paketes eruiert.

Werden weder Änderungszeitstempel noch Zugriffszeitstempel beim Projekt mitgeliefert, wird das Paket im Graphen *no-files* (keine Dateien) aufgelistet.

2.14.2 Vergleichen von Paketen

Unter dem Debian Popcon Graph [Debian-Popcon-Graph] können Sie dies sogar benutzen, um den Verlauf der Beliebtheit von Paketen gegenüberzustellen. Abbildung 2.10 zeigt beispielhaft einen Vergleich zwischen *screen* und *tmux* in den beiden Metriken *installed* und *vote*.

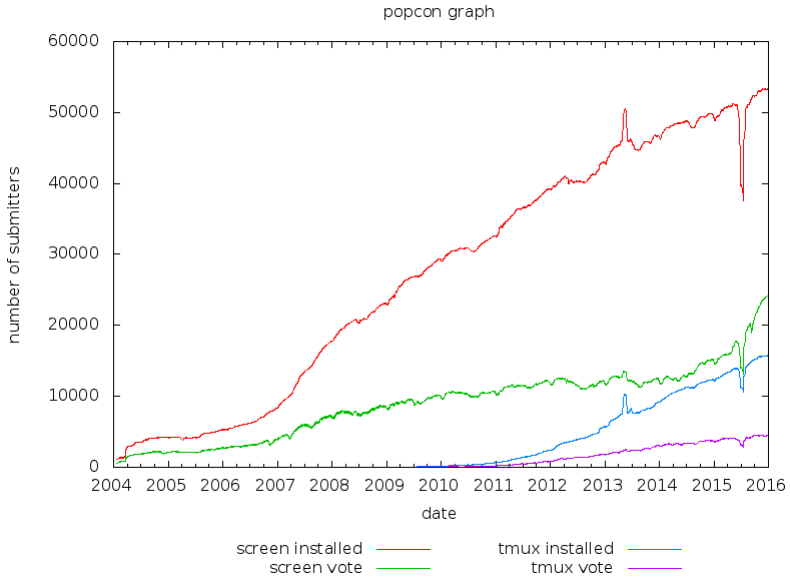

Abbildung 2.10: Vergleich Installationen und Nutzung der Pakete *screen* und *tmux*

2.15 Lokale Paketmarkierungen

Ein installiertes Debianpaket kann zusätzliche, lokale Markierungen haben, die z.B. dessen Aktualisierung beeinflussen oder, wenn kein anderes Paket mehr von ihm abhängt, auch seine automatische Deinstallation veranlassen oder verhindern.

2.15.1 Paketmarkierungen, die von verschiedenen Programmen genutzt werden

Diese Markierungen werden teilweise automatisch von APT, Aptitude und Co. gesetzt, können aber auch manuell gesetzt oder entfernt werden.

automatisch installiert (*automatic*)
: das Paket wurde automatisch installiert, i.d.R. als Abhängigkeit eines anderen Pakets (siehe Abschnitt 8.17). Diese Markierung veranlasst, dass dieses Paket wieder entfernt wird, wenn keine weiteren, installierten Pakete von diesem abhängen.

manuell installiert (*manual*)
: das Paket wurde manuell, d.h. explizit installiert. Diese Markierung verhindert, dass dieses Paket automatisch mit entfernt wird, wenn kein weiteres Paket von ihm abhängt.

halten (*hold*)
: das Paket wird in der vorliegenden, installierten Version auf dem System gehalten und nicht aktualisiert (*upgrade*) oder deinstalliert (siehe Abschnitt 8.39 und Abschnitt 8.41).

Die Markierung *manuell installiert* entspricht defacto dem Nicht-Vorhandensein einer *automatisch installiert* Markierung. Ein Paket hat jeweils immer genau eine der beiden Markierungen *manuell installiert* oder *automatisch installiert*.

Die vorgenannten Paketmarkierungen werden von `dpkg` (nur *hold*), APT und `aptitude` ausgewertet. Die Unterscheidung *automatisch/manuell installiert* wird dazu in der Datei `/var/lib/apt/extended_states` gespeichert, die *hold*-Markierungen in `/var/lib/dpkg/status`[4]

Informationen zu jedem Paket in der Datei /var/lib/apt/extended_states (Ausschnitt)

[4] In früheren Debian-Veröffentlichungen wurden die *hold*-Markierungen von `aptitude` und `dpkg` getrennt gespeichert und `apt-get` wusste nichts von der *hold*-Markierung. Auch wurde die *automatisch installiert*-Markierung zuerst von Aptitude eingeführt und dementsprechend anfangs nur in `/var/lib/aptitude/pkgstates` gespeichert.

```
...

Package: gnome-menus
Auto-Installed: 0
Architecture: i386

Package: libfont-afm-perl
Auto-Installed: 1
Architecture: i386

Package: libhtml-parser-perl
Auto-Installed: 1
Architecture: i386

...
```

Ein Paket "auf hold" in der Datei /var/lib/dpkg/status (Ausschnitt)

```
...

Package: awesome
Status: hold ok installed
Priority: optional
Section: x11
Installed-Size: ...
...
```

2.15.2 Aptitude-spezifische Paketmarkierungen

`aptitude` speichert weitere Informationen zu den Paketen eigenständig in der Datei `/var/lib/aptitude/pkgstates`. Dazu gehören:

Verbotene Versionen (*forbid-version*/*ForbidVer*)
: Vom lokalen Adminstrator nicht erwünschte Version, die nicht installiert werden darf, auf die nicht aktualisiert werden darf bzw. die beim Aktualisieren übersprungen werden soll.

Neue Pakete (*New Packages*/*Unseen*)
: Aptitude pflegt eine Liste mit neu in den Paketlisten der abonnierten APT-Repositories aufgetauchten Paketen, die vom lokalen Adminstrator zurückgesetzt werden kann (`aptitude forget-new`).

Entfernungsgrund (*Remove-Reason*)
: Aptitude zeigt an, warum ein Paket entfernt wird: wegen nicht (mehr) erfüllter Abhängigkeiten, Konflikten mit anderen Paketen, oder weil es nicht gebraucht wird (kein Paket mehr davon abhängt). Wird solch eine Paketentfernung nur vorgemerkt, so speichert Aptitude bis zur Entfernung auch den Grund für diese.

Benutzerspezifische Markierungen (*User Tags*)
: Aptitude erlaubt Benutzern Paketen mit dem Unterkommando `add-user-tag` eigene Markierungen zu setzen, nach denen der Benutzer danach mit dem Muster `?user-tag(...)` auch wieder suchen kann.

Aptitude-spezifische Zusatzinformationen zu Paketen (Ausschnitt)

```
...

Package: python3-pkg-resources
Architecture: amd64
Unseen: no
State: 1
Dselect-State: 1
Remove-Reason: 0
```

```
ForbidVer: 18.8-1
User-Tags: broken-by-807773

...
```

2.15.3 Lesen und Anzeigen einer Markierung mit `aptitude`

Sichtbar werden alle Markierungen zu einem Paket, wenn Sie die Details dazu erfragen – entweder direkt über die Kommandozeile oder in der Textoberfläche zu `aptitude`. Wir verdeutlichen Ihnen das hier anhand des installierten und gehaltenen Pakets *wireshark*.

Auf der Kommandozeile verwenden Sie hierfür `aptitude` mit dem Unterkommando `show` gefolgt vom Paketnamen. In den Zeilen 2 und 3 der nachfolgenden Ausgabe erfahren Sie einerseits, dass *wireshark* zurückgehalten wird und andererseits nicht automatisch installiert wurde.

Darstellung der Markierungen zum Paket python-pkg-resources mittels aptitude

```
$ aptitude show python-pkg-resources
Paket: python-pkg-resources
Zustand: Installiert
Verbotene Version: 18.8-1
Automatisch installiert: ja
Version: 18.7-1
...
Benutzermarkierungen: broken-by-807773

...
$
```

In der Textoberfläche von `aptitude` bekommt jeder Eintrag in der Paketliste zusätzliche Buchstaben. Dabei stehen die Buchstaben `h` für *hold* und `A` für *automatic* (siehe Abbildung 2.11).

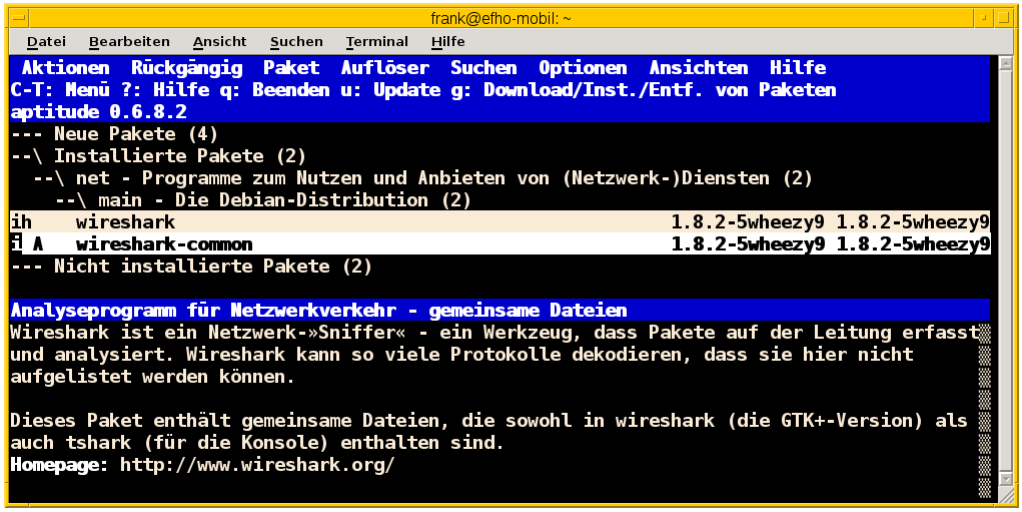

Abbildung 2.11: Ausgabe der Paketmarkierungen in der Textoberfläche von `aptitude`

`aptitude` kann ebenfalls nach allen Paketen fahnden, die automatisch installiert wurden und dazu das Flag *automatic* tragen. Es kennt dazu das spezielle Muster `~M` zum Unterkommando `search`. Ausführlicher besprechen wir das in Abschnitt 8.10.

2.15.4 Lesen und Anzeigen einer Markierung mit `apt-mark`

Das Werkzeug `apt-mark` ist spezialisiert auf die Paketmarkierungen und kann Ihnen die Pakete ausgeben, bei denen nur ein bestimmtes Paketflag gesetzt ist. Es kennt dazu die drei Unterkommandos `showauto`, `showmanual` und `showhold` für alle automatisch oder manuell installierten Pakete bzw. die Pakete, deren Zustand beibehalten wird.

Nachfolgend sehen Sie beispielhaft nur das Ergebnis des Aufrufs für die manuell installierten Pakete. Auf automatisch installierte Pakete gehen wir genauer in Abschnitt 8.10 ein. Dem Umgang mit dem *hold*-Flag in der Praxis ist Kapitel 16 gewidmet.

Auflistung aller manuell installierten Pakete mittels apt-mark

```
# apt-mark showmanual
abiword
acpi
acpi-support
acpi-support-base
...
#
```

Liste der Pakete eingrenzen, die überprüft werden
Geben Sie beim Aufruf keine weiteren Parameter an, werden alle Pakete geprüft. Übergeben Sie hingegen eine eigene Paketliste als Datei, untersucht `apt-mark` die darin genannten Pakete auf das Vorhandensein des jeweiligen Paketmarkierungen.

2.15.5 Setzen und Entfernen einer Markierung mit apt-mark

Die Markierungen *automatic* und *manual* werden von den Programmen zur Paketverwaltung eigenständig gesetzt, wenn Sie Pakete installieren. Grundlage sind die ausgewerteten Paketabhängigkeiten. Trotzdem können Sie stets eigenhändig eingreifen, sofern dazu Ihrerseits Bedarf besteht.

`apt-mark` kennt dafür die drei Schalter `auto` für automatisch, `manual` für manuell und `hold` für gehalten, mit dem Sie die entsprechende Markierung für ein angegebenes Paket explizit setzen können. Dazu erwartet `apt-mark` als Parameter ein einzelnes Paket oder eine Paketliste. Die nachfolgende Ausgabe zeigt das Setzen der Markierung *manual* für das Paket *wireshark*.

Setzen der Paketmarkierungen manual für das Paket wireshark

```
# apt-mark manual wireshark
wireshark wurde als manuell installiert festgelegt.
#
```

Für das Halten eines Pakets existieren die Unterkommandos `hold` und `unhold`. Welchen konkreten Nutzen das haben kann, erfahren Sie unter „Ausgewählte Pakete nicht aktualisieren“ in Kapitel 16.

2.15.6 Was passiert, wenn Paketmarkierungen geändert werden?

Durch das Setzen von Paketmarkierungen verändert sich die Art wie Paketabhängigkeiten ausgewerten werden und damit die Vorschläge durch die Paketverwaltung. `dpkg`, `apt`, `apt-get` und `aptitude` respektieren die von Ihnen gesetzten Markierungen. `apt`, `apt-get` und `aptitude` empfehlen Ihnen bei einer Änderung des Paketbestands beispielsweise andere Pakete als sonst, um die Paketabhängigkeiten nicht zu verletzen. Oder sie schlagen vor, bestimmte Pakete zu entfernen, da sie neu als nicht mehr gebraucht angesehen werden.

Setzen oder Entfernen Sie bewusst das *hold*-Flag und legen somit eine Version explizit fest, nehmen Sie Einfluss auf den Zustand Ihres Systems. Wobei Ihnen das von Nutzen sein kann, erklären wir unter „Ausgewählte Pakete nicht aktualisieren“ (Kapitel 16) ausführlicher.

2.15.7 Setzen und Entfernen einer Markierung mit aptitude

Alternativ dazu kann man auch `aptitude` verwenden. Dort heissen nicht nur die Unterkommandos teilweise etwas anders, `aptitude` in der Standardeinstellung will neu mangels Abhängigkeiten nicht mehr benötigte Pakete auch direkt entfernen. Im u.g. Beispiel gibt es z.B. Pakete, die eine Abhängigkeit auf das Paket *wireshark* haben, aber keine, die eine Abhängigkeit auf *zshdb* haben. Entsprechend will `aptitude` es auch direkt entfernen.

Setzen von Paketmarkierungen mit aptitude

```
# aptitude markauto wireshark zshdb
Die folgenden Pakete werden ENTFERNT:
  zshdb{u}
0 Pakete aktualisiert, 0 zusätzlich installiert, 1 werden entfernt und 26 nicht  ↩
    aktualisiert.
0 B an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 451 kB frei werden ↩
    .
Möchten Sie fortsetzen? [Y/n/?] n
Abbruch.
# aptitude unmarkauto wireshark zshdb
Es werden keine Pakete installiert, aktualisiert oder entfernt.
0 Pakete aktualisiert, 0 zusätzlich installiert, 0 werden entfernt und 26 nicht  ↩
    aktualisiert.
0 B an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 0 B zusätzlich  ↩
    belegt sein.
#
```

Ebenfalls fällt auf, das `aptitude` im Gegensatz zu `apt-mark` nicht angibt, dass sich eine Markierung geändert oder nicht geändert hat, sondern, dass es keine Pakete entfernen oder aktualisieren will.

Allerdings aktualisiert es (in der Standardeinstellung) nicht automatisch Pakete, bei denen die *hold*-Markierung entfernt wurde:

Setzen eines Paketes auf hold mit aptitude

```
# aptitude search '~U'
i A awesome                          - Hochkonfigurierbarer Fenstermanager für X
# aptitude hold awesome
Es werden keine Pakete installiert, aktualisiert oder entfernt.
0 Pakete aktualisiert, 0 zusätzlich installiert, 0 werden entfernt und 26 nicht  ↩
    aktualisiert.
0 B an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 0 B zusätzlich  ↩
    belegt sein.
# aptitude search '~U'
ihA awesome                          - Hochkonfigurierbarer Fenstermanager für X
# aptitude unhold awesome
Es werden keine Pakete installiert, aktualisiert oder entfernt.
0 Pakete aktualisiert, 0 zusätzlich installiert, 0 werden entfernt und 26 nicht  ↩
    aktualisiert.
0 B an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 0 B zusätzlich  ↩
    belegt sein.
# aptitude search '~U'
i A awesome                          - Hochkonfigurierbarer Fenstermanager für X
#
```

2.16 Wie finde ich passende Pakete

2.16.1 Paketquellen

Debianpakete sind von verschiedenen Orten und Medien verfügbar. Dazu zählen sowohl Online- als auch Offline-Quellen, bspw. offizielle, private und unternehmenseigene Repositories und Spiegelserver (*Mirrors*). Für die Recherche und Installation ohne

Internetanbindung stehen bspw. vorbereitete Distributionsimages in unterschiedlichen Größen und Zusammenstellungen für CD, DVD, Blu-ray und USB-Stick über die Webseite des Debian-Projekts bereit [Debian-besorgen].

Je nach den persönlichen Vorlieben sowie der Bandbreite der lokalen Internetanbindung ist jeweils die eine oder andere Variante zur Installation empfehlenswert – eine pauschale Empfehlung können wir Ihnen an dieser Stelle leider nicht geben. Für eine Erstinstallation hat sich bei uns die Reihenfolge *Bezug und Installation über ein kleines Installationsimage* (genannt *Netinst-ISO*) und die nachfolgende, individuelle Auswahl der benötigten Programme über eine Netzwerkinstallation vielfach bewährt. Damit bleiben die eingerichteten Debian-Systeme von Beginn an überschaubar und pflegeleicht und enthalten möglichst wenig Ballast.

Die Auswahl eines Spiegelservers, der zu Ihren technischen Gegebenheiten und Gewohnheiten in der Benutzung Ihres Debian-Systems passt, ist eine Philosophie für sich. Auf die unterschiedlichen Varianten für bereits bestehende Spiegelserver gehen wir genauer in Abschnitt 3.4 ein. Was Sie tun müssen, um hingegen einen eigenen Spiegelserver aufzusetzen und zu betreiben, geht über das Basiswissen deutlich hinaus. Wir erklären Ihnen die Vorgehensweise dazu in Kapitel 28.

2.16.2 Paketnamen

Ist Ihnen der Name eines Pakets oder ein Fragment daraus bekannt, stehen Ihnen alle Möglichkeiten offen. Einerseits helfen Ihnen die Werkzeuge `dpkg`, `apt-cache` sowie `aptitude` auf der Kommandozeile weiter. Desweiteren verfügen die graphischen Programme wie beispielsweise Synaptic (siehe Abschnitt 6.4.1), SmartPM (siehe Abschnitt 6.4.3) oder auch PackageKit (siehe Abschnitt 6.4.5) über eine entsprechende Suchfunktion. Für eine Recherche über das Internet hilft Ihnen nicht nur die Webseite des Debian-Projekts weiter, sondern auch spezielle Suchmaschinen und Verzeichnisdienste. Alle genannten Varianten stellen wir Ihnen unter Abschnitt 8.19 genauer vor.

2.16.3 Paketeigenschaften und Einordnung

Bei den oben angesprochenen Varianten können Sie neben der Einordnung in die jeweilige Paketkategorie (siehe Abschnitt 2.8) bspw. auch über die Veröffentlichungen (siehe Abschnitt 2.10), den Maintainer (siehe Abschnitt 8.21), den Paketinhalt (siehe Abschnitt 8.22) oder ein Fragment aus dem Paket (siehe Abschnitt 8.24) suchen. Darüber hinaus gibt es eine konzept- und facettenbasierte Suche mit Hilfe von Debtags. Letzteres besprechen wir detailliert unter „Erweiterte Paketklassifikation mit Debtags“ (siehe Kapitel 13).

Teil II

Werkzeuge

Kapitel 3

Paketquellen und Werkzeuge

3.1 Paketquellen

3.1.1 Begriff und Hintergrund

Eine Paketquelle bezeichnet einen Ort, von dem aus Softwarepakete zur Verfügung stehen. Alternativ und gleichbedeutend werden dafür auch die Begriffe *APT-Repository*, *Repository* oder ganz kurz nur *Repo* benutzt. Der Begriff *Paketmirror* – oder auch komplett eingedeutscht als *Paketspiegel* – wird ebenfalls gerne verwendet. Letzteres impliziert aber zusätzlich, dass es sich dabei um eine vollständige Kopie einer Paketquelle handelt, also z.B. um einen offiziellen Spiegelserver von Debian oder Ubuntu.

Eine Paketquelle kann dabei aber auch ein externes Speichermedium wie eine CD, DVD, Blu-ray, eine Speicherkarte oder ein USB-Stick sein, aber auch ein lokales oder über das Netzwerk angebundenes Verzeichnis auf einer Festplatte. Waren noch vor wenigen Jahren die erstgenannten, festen Installationsmedien üblich, werden heute als Paketquelle aufgrund der weitestgehend flächendeckenden Verfügbarkeit des Internets stattdessen FTP- und HTTP-Server bevorzugt. Damit sind die von Ihnen genutzten Paketquellen stets aktuell.

3.1.2 Benutzte Paketquellen

Welche Paketquellen Sie verwenden, legen Sie bei Debian in der Datei `/etc/apt/sources.list` (oder alternativ in auf `*.list` endende Dateien im Verzeichnis `/etc/apt/sources.list.d/`) fest. Diese Dateien zählen damit zu den zentralen Komponenten des Debian-Paketsystems. An diesen Einträgen orientieren sich die Werkzeuge zur Paketverwaltung, wenn es um Änderungen im lokalen Paketbestand und entsprechende Aktualisierungen der Pakete auf Ihrem System geht.

Bei der Auswahl der Paketquellen sind Sie nicht auf lediglich eine dieser o.g. Ressourcen beschränkt. Sie können diese beliebig mischen und somit auch Konzepte zur Ausfallsicherung umsetzen. Diese Konstellation kommt genau dann zum Tragen, wenn Ihre primäre Paketquelle nicht in der gewohnten Art und Weise zur Verfügung steht, bspw. bei einem Ausfall des Internetzugangs oder der Wartung des bevorzugten Paketspiegels.

3.1.3 Aufbau und Struktur einer Paketquelle

Jede Paketquelle folgt einer festgelegten Verzeichnisstruktur [Aoki-Debian-Referenz], auf die sich die einzelnen Programme zur Paketverwaltung stützen. Interessant wird diese Struktur genau dann, wenn Sie eine Paketquelle mit selbsterstellten Paketen oder einen eigenen Paketmirror aufsetzen und betreiben möchten (siehe Kapitel 28).

3.2 Empfehlung zum Ablauf für das Hinzufügen und Ändern von Paketquellen

Wie bereits in Abschnitt 3.1 ausgeführt, sind die Datei `/etc/apt/sources.list` und das Verzeichnis `/etc/apt/sources.list.d/` Dreh- und Angelpunkte für alle verwendeten Paketquellen. Erfolgen Änderungen darin, muss die Paketverwaltung anschließend noch über diese Modifikation informiert werden, damit sie den Paketcache anhand der aktualisierten Liste von Repositories auf den neuesten Stand bringt.

Dazu synchronisiert sie die lokal vorliegende Informationen über verfügbare Pakete und deren Abhängigkeiten (siehe Kapitel 7) mit den konfigurierten Paketquellen (Abschnitt 3.1).

Wir empfehlen Ihnen hierfür zu folgendem Ablauf:

1. Erstellen Sie zuerst eine Sicherheitskopie der entsprechenden Datei, z.B. `cp -pv /etc/apt/sources.list /etc/apt/sourceels.list.backup`. Gegebenenfalls macht das auch Ihr Texteditor automatisch.
2. Tragen Sie die neuen oder veränderten Paketquellen in `/etc/apt/sources.list` nach und speichern diese Datei ab. Wenn Sie nur neue Paketquellen hinzufügen wollen, können Sie alternativ auch eine neue Datei im Verzeichnis `/etc/apt/sources.list.d/` anlegen. Der Name dieser Datei muss dann auf `.list` enden.
3. Sofern dies *keine* offiziellen Debian-Repositories sind, verifizieren Sie die Paketquelle, die Sie hinzugefügt oder geändert haben. Unter „Paketquelle auf Echtheit überprüfen“ in Abschnitt 3.12 erfahren Sie, wie das zu erfolgen hat. Offizielle Paketquellen sollten alle mit den bereits mitgelieferten Schlüsseln ihrer Debian-Installation verifiziert werden können.
4. Aktualisieren Sie die lokalen Paketlisten mit einem der Kommandos `apt-get update`, `aptitude update` oder ab Debian 8 *Jessie* auch mit `apt update`. Bitte beachten Sie dazu auch unsere Anmerkungen unter „Liste der verfügbaren Pakete aktualisieren“ in Abschnitt 3.13. Handelt es sich um ein Upgrade auf eine neue Version Ihrer Distribution, lesen Sie bitte dazu zusätzlich unter „Distribution aktualisieren“ in Abschnitt 8.45 nach.

Mit dieser Vorgehensweise ist sichergestellt, dass die Paketverwaltung Ihre Veränderungen in der Liste der Paketquellen beachtet hat. Nun können Sie die Pakete aus den geänderten oder neuen Paketquellen zu Ihrem System hinzufügen.

Mit etwas Automatisierung den Ablauf vereinfachen
Möchten Sie diese Schrittfolge automatisieren, hilft Ihnen das Kommando `add-apt-repository` weiter. Dessen Möglichkeiten besprechen wir genauer in Abschnitt 3.9.

Im Bedarfsfall können Sie auch auf den Stand vor Ihren Veränderungen zurückgreifen. Sollte dies erforderlich sein, restaurieren Sie die im ersten Schritt angelegte Sicherheitskopie oder — falls Sie nur eine neue Datei im Verzeichnis `/etc/apt/sources.list.d/` angelegt haben, löschen Sie diese — und führen das Kommando `apt-get update` respektive `aptitude update` erneut aus. Ab Debian 8 *Jessie* ist auch der Aufruf `apt update` zulässig.

Versionierung statt manuellem Backup
Anstatt manuell Backups zu machen, können Sie auch das Verzeichnis `/etc/apt/` mit einer Versionsverwaltung wie z.B. Git versionieren. Das Debian-Paket *etckeeper* [Debian-Paket-etckeeper] bietet dies sogar automatisiert bei jeder Paketinstallation, -Aktualisierung oder -Entfernung an, versioniert dann aber gleich das ganze Verzeichnis `/etc/`.

3.3 `/etc/apt/sources.list` verstehen

3.3.1 Format der Paketliste

Wie auf UNIX/Linux-Systemen üblich, ist die Datei `/etc/apt/sources.list` eine reine Textdatei. Die Einträge darin erfolgen zeilenweise. Jede einzelne Paketquelle beschreiben Sie vollständig in einer separaten Zeile.

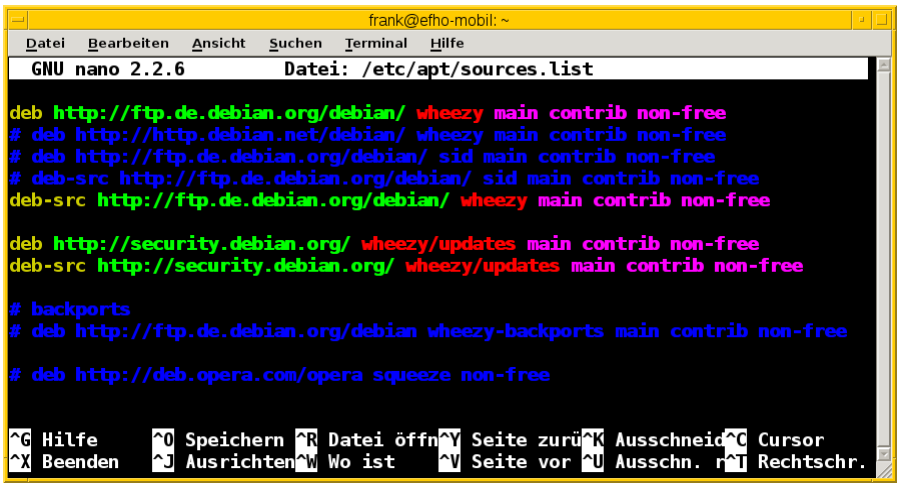

Abbildung 3.1: `/etc/apt/sources.list` im Texteditor *nano*

Änderungen nehmen Sie mit Hilfe eines beliebigen Texteditors als Benutzer `root` vor, bspw. mittels *vim*, *emacs* oder *nano* (siehe Abbildung 3.1). Haben Sie in Ihrem Texteditor die Syntaxhervorhebung aktiviert, erfassen Sie die Struktur der einzelnen Einträge leichter und bemerken sofort fehlerhafte Änderungen durch eine entsprechende Einfärbung des Textes.

Sie fügen eine weitere Paketquelle hinzu, indem Sie die Liste um eine weitere Zeile ergänzen. Tragen Sie dazu in einer freien oder zusätzlichen Zeile die gewünschte Paketquelle nach. Um eine bereits erfasste Paketquelle zu modifizieren, ändern Sie den entsprechenden Listeneintrag. Mit Hilfe des #-Zeichens zu Beginn einer Zeile kommentieren Sie den jeweiligen Eintrag aus. Eine Paketquelle entfernen Sie endgültig aus der Liste, indem Sie die betreffende Zeile löschen.

Anzahl der Einträge
Es gibt keine Begrenzung für die Anzahl der Einträge. Bitte beachten Sie aber, dass die Zeit und das Übertragungsvolumen für die Aktualisierung der Paketlisten umso größer wird, je mehr Einträge vorhanden sind.

Bei der späteren Aktualisierung der lokalen Paketliste mittels `apt-get update` oder `aptitude update` (siehe Abschnitt 8.39) werden die Paketquellen in der Reihenfolge abgearbeitet, wie sie in der Datei `/etc/apt/sources.list` aufgeführt sind. Ignoriert werden dabei Leerzeilen und die Einträge, die mit einem Hashzeichen # beginnen und somit auskommentiert sind.

Empfehlung zur Abfolge
Für das Hinzufügen und Ändern der Paketquellen empfehlen wir Ihnen eine bestimmte Reihenfolge (siehe Abschnitt 3.2). Damit erleben Sie zukünftig keine bösen Überraschungen mehr.

Zur Automatisierung des Vorgangs wurden ebenfalls eine Reihe von Programmen entwickelt. Dazu zählen `apt-cdrom` (siehe dazu Abschnitt 3.8), `add-apt-repository` (siehe dazu Abschnitt 3.9) und `apt-spy` (siehe dazu Abschnitt 3.5.2). Sind Sie hingegen weniger tastaturaffin, bieten sich als weitere Möglichkeiten sowohl Synaptic, das Ubuntu Software Center sowie der Sources List Generator für Debian und Ubuntu an. Diese Programme stellen wir Ihnen in Abschnitt 3.10 und Abschnitt 3.11 ausführlicher vor.

3.3.2 Format eines Eintrags

Jeder Eintrag in der Datei `/etc/apt/sources.list` folgt einem festen Muster mit einer genauen Abfolge von definierten Feldern:

```
Art_der_Quelle URI Distribution [Komponente 1] [Komponente 2] [...]
```

Jedes dieser Felder hat eine bestimmte Funktion und erlaubt nur ausgewählte Inhalte:

Art der Quelle
bezeichnet den verwendeten Pakettyp, `deb` für Debian-Binärpakete und `deb-src` für Debian-Quellpakete. Genauer gehen wir dazu unter „Debians Paketvarianten“ in Abschnitt 2.7 und „Debian-Paketformat“ im Detail in Kapitel 4 ein.

URI
legt die Art der Installationsquelle fest. Hierbei stehen:

- `file`: Installationsquelle ist ein Verzeichnis. Dieses kann sowohl lokal vorliegen, als auch von extern eingebunden sein, bspw. über ein Netzwerkdateisystem wie AFS, NFS oder SMB
- `cdrom`: CD, DVD oder Blu-ray
- `http`: HTTP-Server
- `ftp`: FTP-Server
- `copy`: identisch zu `file`, aber die bezogenen Debianpakete werden zusätzlich im lokalen Verzeichnis `/var/cache/apt/archives/` abgelegt
- `mirror`: Auswahl einer Installationsquelle anhand der GeoIP des Servers (siehe Abschnitt 3.6.2)
- `debtorrent`: die Pakete werden über DebTorrent auf der Basis des BitTorrent-Protokolls bezogen [Debian-Wiki-DebTorrent]. Basis dafür ist das Debianpaket *apt-transport-debtorrent* [Debian-Paket-apt-transport-debtorrent].

Distribution
benennt die Veröffentlichung (siehe Abschnitt 2.10), aus der Pakete installiert werden sollen. Typisch ist hier die Angabe des Entwicklungsstands (siehe Abschnitt 2.10.1) wie bspw. *stable*, *unstable* oder *testing* sowie die Nennung des alternativen Distributionsnamens wie bspw. *Wheezy*, *Jessie* oder *Sid* (siehe Abschnitt 2.10.2).

Bitte beachten Sie bei Debian und Ubuntu die Kleinschreibung des Namens. Nicht-offizielle Paketquellen können an dieser Stelle jedoch auch sonstige Zeichenketten bis hin zu einem . verlangen.

Komponente
bestimmt den Distributionsbereich, d.h. bspw. bei Debian *main*, *contrib* oder *non-free*. Ausführlicher gehen wir darauf in Abschnitt 2.9 ein.

3.3.3 Beispieleinträge für offizielle Pakete

Der Standardeintrag für den Bezug von stabilen Debianpaketen aus dem Bereich *main* mit dem deutschen Spiegelserver als Paketquelle sieht folgendermaßen aus:

```
deb http://ftp.de.debian.org/debian/ stable main
```

Mit diesem Eintrag beziehen Sie stets nur Pakete aus der aktuellen, stabilen Veröffentlichung. Erscheint eine neue Veröffentlichung, sind Sie damit auf der sicheren Seite und wechseln automatisch zum Nachfolger.

Tragen Sie hingegen anstatt von *stable* den entsprechenden Aliasnamen der Veröffentlichung in Kleinbuchstaben wie bspw. *wheezy* oder *jessie* ein, nutzen Sie ausschließlich Pakete aus der damit spezifizierten Veröffentlichung, die diesen Aliasnamen trägt. Möchten Sie später von dieser auf eine andere Veröffentlichung wechseln, passen Sie zunächst den Aliasnamen im Eintrag entsprechend an und aktualisieren nachfolgend die lokale Paketdatenbank (siehe „Distribution aktualisieren“ in Abschnitt 8.45).

Um hingegen zusätzlich die Pakete aus weiteren Paketbereichen wie bspw. *contrib* und *non-free* zu verwenden, ändern Sie den Eintrag auf das Folgende, hier wiederum mit expliziter Angabe des Aliasnamens *jessie*. In welcher Reihenfolge Sie die Paketbereiche angeben, spielt keine Rolle. Üblich ist jedoch die Abfolge anhand des Freiheitsgrades der Softwarelizenz in der Form von *main contrib non-free*.

```
deb http://ftp.de.debian.org/debian/ jessie main contrib non-free
```

Auswahl eines Paketmirrors
Mehr Informationen zur Auswahl eines für Sie am besten geeigneten Paketmirrors erfahren Sie unter „Geeigneten Paketmirror auswählen“ in Abschnitt 3.4. Mit dieser Angabe können Sie die Bezugszeiten für Aktualisierungen der Paketlisten und der Pakete erheblich zu ihren Gunsten beeinflussen.

3.3.4 Verzeichnis als Paketquelle

Pakete können Sie auch aus einem Verzeichnis ihres Debian-Systems integrieren. Dabei sind Sie nicht auf lokale Einträge beschränkt, sondern können auch auf entfernte Ressourcen zugreifen, bspw. ein NFS- oder SMB-Share. Voraussetzung ist allerdings, dass die angegebene Ressource vorab in den Verzeichnisbaum eingehängt wurde (auf engl. *mounted*) und APT darauf zugreifen darf. Eine lokale Ressource geben Sie über das Schlüsselwort `file` an, hier am Beispiel des Verzeichnisses `/home/benutzer/debian`:

```
deb file:/home/benutzer/debian stable main contrib non-free
```

Ein Eintrag für einen externen Datenträger, bspw. eine CD, DVD oder Blu-ray, sieht ähnlich wie die vorhergehenden Beispiele aus. Nach dem Schlüsselwort `deb` folgt der Wert `cdrom` mit der Kennung des Datenträgers zur Installation. Am Schluss des Eintrags finden Sie die Veröffentlichung und den Distributionsbereich. Nachfolgend sehen Sie einen Eintrag für eine CD, auf dem Ubuntu 12.04 LTS *Precise Pangolin* enthalten ist:

```
deb cdrom:[Ubuntu 12.04 LTS _Precise Pangolin_ - Release i386 (20120423)]/ precise main ↩
    restricted
```

Automatisierung der Eintragung
Obige Einträge können Sie von Hand vornehmen. Das Werkzeug `apt-cdrom` vereinfacht den Vorgang jedoch erheblich. Unter „Physische Installationsmedien mit `apt-cdrom` einbinden“ in Abschnitt 3.8 besprechen wir das Programm genauer.

3.3.5 Einträge für Sicherheitsaktualisierungen

Häufig, aber in unregelmäßigen Abständen – d.h. wenn es erforderlich ist – kündigt das Debian Security-Team [Debian-Security] Sicherheitsaktualisierungen an und stellt diese bereit. Um von diesen Aktualisierungen zu profitieren, ergänzen Sie die Datei `/etc/apt/sources.list` um den folgenden Eintrag:

```
deb http://security.debian.org/ jessie/updates main contrib non-free
```

Obige Angabe beinhaltet beispielhaft wiederum die explizite Angabe des Aliasnamens der Veröffentlichung Debian 8 *Jessie*. Dieser Name wird gefolgt vom Unterverzeichnis `updates` und den daraus gewünschten Distributionsbereichen *main*, *contrib* und *non-free*.

3.3.6 Einträge für zusätzliche, nicht-offizielle Pakete

Nicht alle verfügbaren Softwareveröffentlichungen werden in die offiziellen Paketquellen von Debian aufgenommen. Viele Projekte stellen Programmversionen als `deb`-Pakete bereit, die sich von der Version her von der stabilen Veröffentlichung von Debian unterscheiden.

Im folgenden Beispiel sehen Sie die Einbindung der Paketquellen des PostgreSQL-Projekts [APT-Repo-PostgreSQL] und des X2Go-Projekts [APT-Repo-X2Go] für Debian 7 *Wheezy*:

```
deb http://apt.postgresql.org/pub/repos/apt/ wheezy-pgdg main
deb http://packages.x2go.org/debian wheezy main
```

Ähnliches gilt für Unternehmen, die erfreulicherweise inzwischen vielfach eigene `deb`-Pakete für ihre Produkte zur Verfügung stellen. Die exakte Bezugsquelle finden Sie zumeist auf der Webseite des jeweiligen Unternehmens. Um bspw. die Pakete für den Webbrowser Opera des gleichnamigen skandinavischen Herstellers einzubinden, hilft Ihnen folgender Verweis[1] auf den Bereich *non-free* auf dessen Paketserver:

```
deb http://deb.opera.com/opera stable non-free
```

[1] Die aktuelle Konfiguration des APT-Repositories erlaubt nur die Verwendung von *stable* als Veröffentlichung. Verwenden Sie z.B. *jessie* anstatt von *stable*, so beschwert sich APT, dass dies nicht vorgesehen sei.

Ergänzung der Signatur der Paketquelle
Damit Debian dieser zusätzlichen Paketquelle auch vertraut, überprüft es dazu eine entsprechende digitale Signatur. Wie dieses Konzept funktioniert und Sie einen passenden Schlüssel beziehen, lesen Sie unter „Paketquelle auf Echtheit überprüfen“ in Abschnitt 3.12.

Eigene .list-Datei für fremde Paketquellen.
Anstatt alle Einträge direkt in die Datei `/etc/apt/sources.list` zu schreiben, können Sie einen oder mehrere Einträge auch in separate Dateien unter `/etc/apt/sources.list.d/` ablegen. Dateien in diesem Verzeichnis bedürfen der Endung *.list*, um von APT beachtet zu werden.
So könnten Sie z.B. die Beispiele in diesem Abschnitt in den Dateien `/etc/apt/sources.list.d/postgresql.list`, `/etc/apt/sources.list.d/x2go.list` und `/etc/apt/sources.list.d/opera.list` speichern. Damit behalten Sie bereits anhand des Dateinamens den Überblick, aus welchen Fremdquellen weitere Pakete bezogen werden.

3.3.7 Einträge für Quellpakete

Um Debian-Quellpakete (siehe Abschnitt 2.7.4) zu nutzen, benötigen Sie eine weitere Zeile in ihrer Paketliste. Im Vergleich zu Binärpaketen ändert sich lediglich das Schlüsselwort am Anfang eines Eintrags von `deb` auf `deb-src`. Danach erwartet APT wie gewohnt den Eintrag der Paketquelle. Für die offiziellen Quellpakete sieht der Eintrag wie folgt aus, hier am Beispiel des deutschen Paketmirrors für Debian 8 *Jessie*:

```
deb-src http://ftp.de.debian.org/debian/ jessie main
```

3.3.8 Einträge für Deutschland

Liegt ihr Lebens- und Arbeitsmittelpunkt in Deutschland oder Sie beziehen die Pakete von einem Paketmirror, der in Deutschland steht, enthält die Datei typischerweise die folgenden Einträge:

```
deb http://ftp.de.debian.org/debian/ jessie main contrib non-free
deb-src http://ftp.de.debian.org/debian/ jessie main contrib non-free

deb http://security.debian.org/ jessie/updates main contrib non-free
deb-src http://security.debian.org/ jessie/updates main contrib non-free
```

Mit den ersten beiden Zeilen beziehen Sie alle Binär- und Sourcepakete für die Distributionsbereiche *main*, *contrib* und *non-free* für die Veröffentlichung Debian 8 *Jessie* vom primären deutschen Debian-Spiegelserver. Mit den Zeilen drei und vier beziehen Sie zusätzlich die dazugehörigen Sicherheitsaktualisierungen für alle Distributionsbereiche der gleichen Veröffentlichung von der zentralen Stelle security.debian.org.

3.4 Geeigneten Paketmirror auswählen

Zentraler Anlaufpunkt für netzbasierte Installationen sind die offiziellen Paketmirrors – auf deutsch auch Spiegelserver genannt – welche die Debianpakete für Sie bereithalten. Diese Paketmirrors sind weltweit verteilt und werden meist ehrenamtlich von einem Verantwortlichen für den jeweiligen Standort oder im Rahmen seiner administrativen Aufgaben vor Ort betreut. Viele Spiegelserver werden automatisch über neue Pakete informiert und abgeglichen und verfügen somit stets über den aktuelle Paketbestand.

Mit dieser dezentralen Verteilung ist gewährleistet, dass Sie bei einem Ausfall oder der Nichtverfügbarkeit eines Paketmirrors problemlos auf eine adäquate Alternative zurückgreifen können, auch wenn diese netztechnisch etwas weiter von ihrem aktuellen Standort entfernt ist. Für die Infrastruktur des Debian-Projekts heißt das außerdem, dass sich die Anfrage- oder Netzlast auf unterschiedliche Mirrors und deren Standorte verteilt. Für Sie bedeutet das konkret, dass sich neben der Verringerung der

Ausfallwahrscheinlichkeit insbesondere die Bezugszeiten für Debianpakete erheblich verringern, da Sie nicht auf einen einzigen Spiegelserver angewiesen sind. Verwenden Sie einen Spiegelserver in ihrer Nähe, merken Sie das insbesondere dann, wenn größere Aktualisierungen erfolgen, bspw. bei einem Distributionswechsel oder -upgrade (siehe Abschnitt 8.39.3).

Jeder Interessierte kann einen solchen Paketmirror betreiben. Wie Sie diesen einrichten, erfahren Sie unter „Einen eigenen APT-Mirror aufsetzen“ in Kapitel 28. Möchten Sie Ihren eigenen Paketmirror auch öffentlich zugänglich machen, nutzen Sie dazu am besten das bereits dafür vorbereitete Formular auf der Webseite des Debian-Projekts [Debian-Spiegel-Informationen].

3.4.1 Paketmirror bei Debian

Das Debian-Projekt pflegt eine offizielle Liste seiner Paketmirrors [Debian-Spiegel-Liste]. Diese Liste ist in *primäre* und *sekundäre* Mirrors gegliedert.

Primäre Mirrors sind dabei als zentrale, stets aktuelle Bezugspunkte mit hoher Last ausgelegt. Neben einer guten Netzanbindung bieten diese auch eine hohe Verfügbarkeit. Sie werden automatisch aktualisiert, sofern es Änderungen im Debian-Projekt bzw. dessen Paketarchiv gibt. Ein Mirror dieser Kategorie ist nach dem folgenden Namensschema erreichbar:

```
ftp.Länderkennung.debian.org
```

Die Länderkennung richtet sich nach dem ISO-Namensschema. Für Deutschland ist das `de`, für Österreich `at` und für die Schweiz `ch`. Der deutsche primäre Paketmirror ist somit unter `ftp.de.debian.org` erreichbar, der österreichische unter `ftp.at.debian.org` und der schweizerische unter `ftp.ch.debian.org`.

Primäre Debian-Mirrors stehen häufig bei Internetprovidern oder Lehr- und Forschungseinrichtungen. Beispielsweise steht der primäre Schweizer Debian-Paketmirror `ftp.ch.debian.org` an der ETH Zürich und einer der Debian-Paketmirrors für Deutschland an der TU Dresden.

Sekundäre Mirrors unterscheiden sich dahingehend von primären Mirrors, dass nicht garantiert ist, dass das volle Spektrum an Debianpaketen und Architekturen (siehe Abschnitt 1.2) geboten wird. Hintergrund kann bspw. eine Begrenzung des verfügbaren Speicherplatzes auf dem Server sein. Das bedeutet jedoch nicht zwangsläufig, dass dieser Mirror schlechter erreichbar sein muss. Im deutschsprachigen Raum betreiben entsprechende Paketmirrors bspw. die Uni Erlangen, die TU Graz und SWITCH, das Schweizer Hochleistungsnetzwerk für die Wissenschaft [SWITCH].

Für jeden Paketmirror existiert eine Beschreibung, die diesen genauer klassifiziert. Dazu gehört z.B. die URL, der Aliasname, der Typ des Paketmirrors, die darüber verfügbaren Architekturen (siehe Abschnitt 1.2) sowie die Informationen zum genauen Standort, zum Betreuer bzw. Betreiber und der Anbindung (IPv4 oder IPv6). Nachfolgender Auszug zeigt die Details für den Paketmirror `ftp.de.debian.org`, der an der TU Dresden beheimatet ist. [2]

Informationen zum Paketmirror ftp.de.debian.org (TU Dresden)

```
Site: ftp.de.debian.org
Alias: ftp1.de.debian.org
Alias: debian.inf.tu-dresden.de
Type: Push-Primary
Archive-architecture: amd64 armel armhf hurd-i386 i386 ia64 kfreebsd-amd64 kfreebsd-i386 ↩
    mips mipsel powerpc s390 s390x sparc
Archive-ftp: /debian/
Archive-http: /debian/
Archive-rsync: debian/
Archive-upstream: ftp-master.debian.org
Archive-method: push
Backports-ftp: /debian-backports/
Backports-http: /debian-backports/
Backports-rsync: debian-backports/
Backports-upstream: syncproxy3.eu.debian.org
Backports-method: push
CDImage-ftp: /debian-cd/
CDImage-http: /debian-cd/
CDImage-rsync: debian-cd/
```

[2] Die Auswahl des Mirrors erfolgte aus zwei Gründen – erstens tief verwurzeltem Lokalpatriotismus von Frank, und zweitens aus dem angebotenen Leistungsumfang heraus. Von diesem Mirror bekommen Sie das ganze Debian-Spektrum.

```
Old-ftp: /debian-archive/
Old-http: /debian-archive/
Old-rsync: debian-archive/
Volatile-ftp: /debian-volatile/
Volatile-http: /debian-volatile/
Volatile-rsync: debian-volatile/
Volatile-upstream: kassia.debian.org
Ports-architecture: alpha arm64 hppa m68k powerpcspe ppc64 sh4 sparc64 x32
Ports-ftp: /debian-ports/
Ports-http: /debian-ports/
Ports-rsync: debian-ports/
Ports-upstream: ftp.debian-ports.org
Country: DE Germany
Location: Dresden
Sponsor: Technical University of Dresden, Dept. of Computer Science http://www.inf.tu- ↩
    dresden.de/
Comment: DFN
IPv6: no
```

3.4.2 Paketmirror für andere Distributionen

Für die anderen `deb`-basierten Distributionen sieht das ähnlich wie bei Debian aus. Eine aktuelle Liste finden Sie auf der Webseite der jeweiligen Distribution. Beispielhaft zeigen wir Ihnen das für die Distribution Linux Mint.

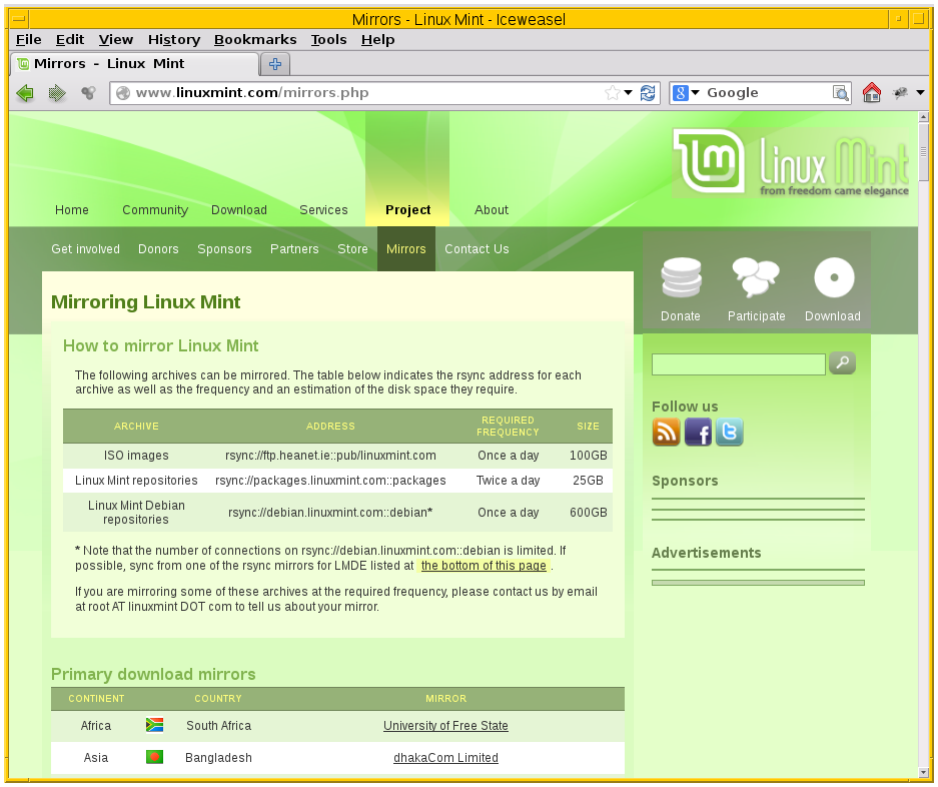

Abbildung 3.2: Auswahl der Paketmirror für Linux Mint

3.4.3 Generischen Mirror verwenden

Wie bereits oben genannt, existieren für Debian *primäre* und *sekundäre* Paketmirrors. Wir empfehlen Ihnen, bei der Auswahl eines Paketmirrors einen solchen zu bevorzugen, der eine möglichst kurze Entfernung zu ihrem Standort hat, mit hoher Verfügbar-

keit glänzt und über eine gute Netzanbindung verfügt. Damit erhöht sich die Zuverlässigkeit ihrer Infrastruktur und insbesondere auch der Komponenten, die von externen Bestandteilen und Diensten abhängig sind.

Sollten Sie nur über einen Zugang mittels Webbrowser verfügen, steht Ihnen die Webseite des Debian-Projekts [Debian-Webseite] zur Verfügung. Darüberhinaus bietet `apt-get.org` [apt-get.org] die Möglichkeit zur Recherche nach einem möglichen inoffiziellen Repository (siehe Abbildung 3.3). Bitte beachten Sie bei der Auswahl der Paketquelle über diesen Dienst, dass nicht jedes der angezeigten Repositories Pakete für alle Architekturen (siehe Abschnitt 1.2) und Veröffentlichungen (siehe Abschnitt 2.10) bereithält.

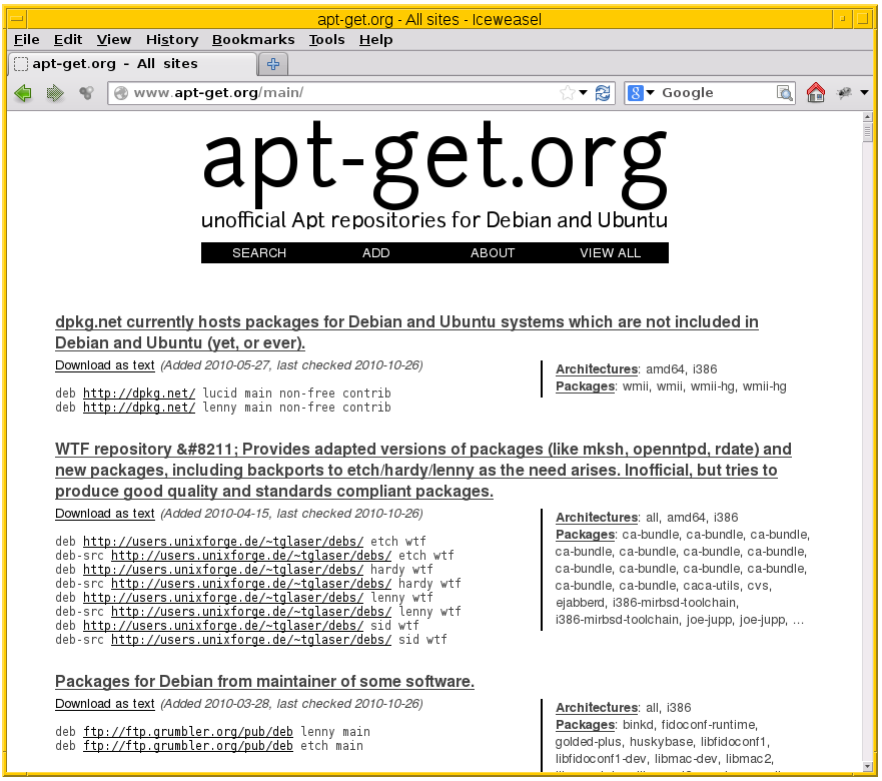

Abbildung 3.3: Auswahl der Paketmirror für bei `apt-get.org`

Sehr hilfreich und zumeist auch der erste Anlaufpunkt für inoffizielle Debianpakete ist die Paketsuche unter dem Menüpunkt Search. Im Eingabefeld geben Sie ein Textfragment aus dem Namen eines Pakets ein, nachdem dann `apt-get.org` seine Liste der Spiegelserver durchforstet. Das Ergebnis ist eine Liste, aus der Sie entnehmen können, von welchem Spiegelserver Sie das gewünschte Paket beziehen können. Neben der Architektur (siehe Abschnitt 1.2) sehen Sie auch die Veröffentlichung (siehe Abschnitt 2.10) und den Distributionsbereich (siehe Abschnitt 2.9), in die das gefundene Paket einsortiert ist. Abbildung 3.4 zeigt das Suchergebnis nach dem Paket *libdvdcss* an, welches bei älteren Veröffentlichungen wie Debian 3 *Woody*, Debian 3.1 *Sarge* oder auch bei *Sid* für die drei Debian-Architekturen *all*, *i386* und *powerpc* zum Lesen von DVDs benötigt wird und hierüber zur Verfügung steht.

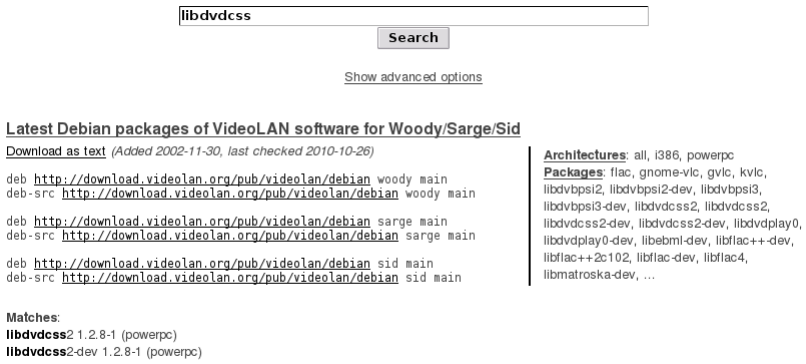

Abbildung 3.4: Suchergebnis der Recherche bei `apt-get.org`

3.5 Am besten erreichbaren Paketmirror finden

Jeder Paketmirror hat eine spezifische Leistungsfähigkeit, die sich an den beiden Kriterien Netzanbindung und Hardwareausstattung messen lässt. Auf diese Merkmale haben Sie zwar nur begrenzt Einfluss, steuern jedoch über die Einträge in der Datei `/etc/apt/sources.list` (siehe Abschnitt 3.3), welchen verfügbaren Paketmirror Sie benutzen.

Anstatt diese Schritte aufwendig über die Kombination einzelner Werkzeuge wie `ping` oder `traceroute` zu ermitteln, sind hier `netselect` [Debian-Paket-netselect] und `netselect-apt` [Debian-Paket-netselect-apt] die besseren Mittel der Wahl. Damit finden Sie den Spiegelserver heraus, der netztechnisch von ihrem aktuellen Standort aus am besten erreichbar ist.

Bis einschließlich Debian 7 *Wheezy* war das Programm `apt-spy` [Debian-Paket-apt-spy] fester Bestandteil der Veröffentlichung. Es wurde nicht in Debian 8 *Jessie* übernommen, steht jedoch weiterhin für *unstable* zur Verfügung.

3.5.1 Paketquellen mit `netselect` nach Pingzeiten und Entfernung auswählen

Die beiden Programme `netselect` und `netselect-apt` überprüfen den von Ihnen benannten Spiegelserver anhand von mehreren Kriterien. Dazu gehört primär die grundsätzliche Erreichbarkeit über das Netzwerk, die Pingzeit – d.h. wieviel Zeit benötigt ein Netzwerkpaket vom Paketmirror zu Ihrem Computer – , sowie die Verlustrate der Netzwerkpakete vom Spiegelserver zu Ihnen. Gleichzeitig wird die Anzahl der Zwischenknoten von Ihrem Computer zum Spiegelserver gezählt, auch genannt *Hops*. Bevorzugt werden lokale Paketmirrors, was sich auch im daraus errechneten Zahlenwert niederschlägt. Je kleiner der ermittelte Wert ist, umso besser ist das für Sie.

Der Unterschied zwischen `netselect` und `netselect-apt` ist folgender: `netselect` gibt nur den ermittelten Zahlenwert für den evaluierten Spiegelserver aus, während `netselect-apt` eine Datei namens `sources.list` im aktuellen Verzeichnis erzeugt. Diese beinhaltet die besten gefundenen Spiegelserver und kann von Ihnen sofort als neue Liste der Paketquellen benutzt werden.

Aktualisierung der Liste der Paketquellen
Zu Änderungen an den Paketquellen beachten Sie bitte auch unsere Hinweise unter „`/etc/apt/sources.list` verstehen“ in Abschnitt 3.3. Wir raten Ihnen dazu, die neue Liste der Paketquellen zuerst lokal zu erstellen und danach manuell in das Verzeichnis `/etc/apt/` zu verschieben.

`netselect` und `netselect-apt` akzeptieren beim Aufruf eine Menge verschiedener Schalter und Parameter. Stets anzugeben ist mindestens ein Spiegelserver, der zu testen ist. Geben Sie hingegen eine ganze Liste an, werden alle daraus nacheinander überprüft. Die nachfolgende Ausgabe zeigt das Ergebnis für fünf angefragte Paketmirrors.

Aufruf von netselect mit fünf verschiedenen Paketmirrors

```
# netselect -v ftp.debian.org http.us.debian.org ftp.at.debian.org download.unesp.br ftp. ↩
    debian.org.br
netselect: unknown host ftp.debian.org.br
Running netselect to choose 1 out of 8 addresses.
..........................................................
   73 ftp.debian.org
#
```

Mit dem zusätzlichen Schalter `-v` regeln Sie die Ausführlichkeit der Ausgabe. Ohne den Schalter geben beide Programme nur den Paketmirror aus, der den besten Wert hat, mit `-vv` bzw. `-vvv` oder sogar `-vvvv` entsprechend mehr Details. Weitere Optionen entnehmen Sie bitte der Manpage zu den Programmen.

Etwas ausführlichere Ausgabe zu den Paketmirrors

```
# netselect -vv ftp.debian.org http.us.debian.org ftp.at.debian.org download.unesp.br ftp. ↩
    debian.org.br
netselect: unknown host ftp.debian.org.br
Running netselect to choose 1 out of 8 addresses.
..........................................................
128.61.240.89          141 ms   8 hops   88% ok ( 8/ 9) [  284]
ftp.debian.org          41 ms   8 hops  100% ok (10/10) [   73]
128.30.2.36            118 ms  19 hops  100% ok (10/10) [  342]
64.50.233.100          112 ms  14 hops   66% ok ( 2/ 3) [  403]
64.50.236.52           133 ms  15 hops  100% ok (10/10) [  332]
ftp.at.debian.org       47 ms  13 hops  100% ok (10/10) [  108]
download.unesp.br      314 ms  10 hops   75% ok ( 3/ 4) [  836]
ftp.debian.org.br     9999 ms  30 hops    0% ok
   73 ftp.debian.org
#
```

In der Ausgabe erscheinen die IP-Adresse bzw. der Hostname (Spalte 1), nachdem aufgelöst wird, die durchschnittliche Paketlaufzeit (Spalte 2), die Anzahl der Zwischenknoten (Spalte 3) sowie die Verlustrate der Pakete auf dem Transportweg (Spalte 4 bis 6). Die Angabe `ok` besagt dabei, dass der Paketmirror über das Netz erreichbar ist. Die Angabe `9999ms` für die Paketlaufzeit besagt hingegen, dass der Paketmirror zum Testzeitpunkt leider nicht erreichbar war.

Die Werte in den runden Klammern in Spalte 6 zeigen, wie der Prozentwert der Verlustrate der Pakete in Spalte 4 zustandekam. Dieser basiert auf der Anzahl Pakete, die der Paketmirror als empfangen bestätigt hat, jeweils gegenübergestellt der Anzahl gesendeter Pakete. Die Zahl in den eckigen Klammern am Ende jeder ausgegebenen Zeile (Spalte 7) ist der Wert, den `netselect` für den jeweiligen Paketmirror ermittelt hat.

Noch mehr Informationen zu den Paketmirrors

```
# netselect -vvv ftp.debian.org http.us.debian.org ftp.at.debian.org download.unesp.br ftp. ↩
    debian.org.br
netselect: unknown host ftp.debian.org.br
Running netselect to choose 1 out of 8 addresses.
128.30.2.36            122 ms   15 hops - HIGHER
64.50.233.100          112 ms   15 hops - OK
ftp.at.debian.org       49 ms   15 hops - OK
min_lag is now 49
64.50.236.52           140 ms   15 hops - OK
ftp.debian.org          42 ms   15 hops - OK
min_lag is now 42
ftp.at.debian.org       48 ms    8 hops - HIGHER
128.30.2.36            117 ms   23 hops - OK
ftp.debian.org          41 ms    8 hops - OK
min_lag is now 41
64.50.233.100          112 ms    8 hops - HIGHER
64.50.236.52           112 ms    8 hops - HIGHER
ftp.debian.org          28 ms    4 hops - HIGHER
ftp.at.debian.org       49 ms   12 hops - HIGHER
ftp.debian.org          38 ms    6 hops - HIGHER
```

```
ftp.at.debian.org       48 ms   14 hops - OK
128.30.2.36            119 ms   19 hops - OK
64.50.233.100          113 ms   12 hops - HIGHER
ftp.debian.org          53 ms    7 hops - HIGHER
ftp.at.debian.org       49 ms   13 hops - OK
64.50.236.52           114 ms   12 hops - HIGHER
ftp.debian.org          42 ms    8 hops - OK
download.unesp.br      306 ms   15 hops - OK
ftp.at.debian.org       48 ms   13 hops - OK
ftp.debian.org          42 ms    8 hops - OK
ftp.at.debian.org       49 ms   13 hops - OK
64.50.233.100          114 ms   14 hops - OK
128.30.2.36            118 ms   17 hops - HIGHER
ftp.debian.org          42 ms    8 hops - OK
64.50.236.52           138 ms   14 hops - HIGHER
ftp.at.debian.org       49 ms   13 hops - OK
ftp.debian.org          41 ms    8 hops - OK
ftp.at.debian.org       49 ms   13 hops - OK
ftp.debian.org          41 ms    8 hops - OK
128.30.2.36            119 ms   18 hops - HIGHER
ftp.debian.org          43 ms    8 hops - OK
ftp.at.debian.org       48 ms   13 hops - OK
64.50.236.52           132 ms   15 hops - OK
ftp.debian.org          43 ms    8 hops - OK
ftp.at.debian.org       48 ms   13 hops - OK
ftp.debian.org          42 ms    8 hops - OK
128.30.2.36            118 ms   19 hops - OK
ftp.at.debian.org       48 ms   13 hops - OK
download.unesp.br      313 ms    8 hops - HIGHER
64.50.236.52           134 ms   15 hops - OK
128.30.2.36            122 ms   19 hops - OK
64.50.236.52           133 ms   15 hops - OK
128.30.2.36            129 ms   19 hops - OK
download.unesp.br      307 ms   12 hops - OK
64.50.236.52           140 ms   15 hops - OK
128.30.2.36            124 ms   19 hops - OK
64.50.236.52           133 ms   15 hops - OK
128.30.2.36            117 ms   19 hops - OK
128.30.2.36            117 ms   19 hops - OK
64.50.236.52           134 ms   15 hops - OK
download.unesp.br      308 ms   10 hops - OK
128.30.2.36            118 ms   19 hops - OK
64.50.236.52           134 ms   15 hops - OK
128.30.2.36            118 ms   19 hops - OK
64.50.236.52           133 ms   15 hops - OK
download.unesp.br      305 ms    9 hops - HIGHER
64.50.236.52           131 ms   15 hops - OK


download.unesp.br      307 ms  10 hops   75% ok ( 3/ 4) [  818]
128.30.2.36            119 ms  19 hops  100% ok (10/10) [  345]
64.50.233.100          113 ms  14 hops   66% ok ( 2/ 3) [  405]
64.50.236.52           134 ms  15 hops  100% ok (10/10) [  335]
128.61.240.89         9999 ms  30 hops    0% ok
ftp.at.debian.org       48 ms  13 hops  100% ok (10/10) [  110]
ftp.debian.org          41 ms   8 hops  100% ok (10/10) [   73]
ftp.debian.org.br     9999 ms  30 hops    0% ok
   73 ftp.debian.org
#
```

Ergebnis des obigen Aufrufs ist eine Empfehlung für einen der Paketmirrors, die Sie im Aufruf benannt haben. Dieser Paketmirror

ist von ihrem Standort aus derzeit am besten erreichbar. Das ermittelte Ergebnis schwankt und hängt stets von der aktuellen Netzauslastung ab.

Die Empfehlung und der ermittelte Zahlenwert stehen in der letzten Zeile der Ausgabe und zeigen hier den Wert 73 für den Server `ftp.debian.org`. Die angegebene Zahl errechnet sich aus den bereits zu Beginn genannten Kriterien und ist vergleichbar mit einem Punktwert, hat jedoch offiziell keine Einheit. Je höher der Wert ist, umso schlechter ist der Paketmirror von Ihrem aktuellen Standort im Netz zu erreichen.

Ist der Paketmirror von einer Firewall geschützt und blockiert UDP-Pakete, kann die Option `-I` von größerem Nutzen sein. Damit werden stattdessen ICMP-Pakete gesendet. Das Ergebnis sehen Sie in der nachfolgenden Ausgabe:

Höchste Stufe der Ausführlichkeit zu den Paketmirrors

```
# netselect -I -vvv ftp.de.debian.org
Running netselect to choose 1 out of 1 address.
ftp.de.debian.org            37 ms   15 hops - OK
min_lag is now 37
ftp.de.debian.org            36 ms    8 hops - OK
min_lag is now 36
ftp.de.debian.org            27 ms    4 hops - HIGHER
ftp.de.debian.org            36 ms    6 hops - HIGHER
ftp.de.debian.org            36 ms    7 hops - OK
ftp.de.debian.org            36 ms    7 hops - OK
ftp.de.debian.org            36 ms    7 hops - OK
ftp.de.debian.org            36 ms    7 hops - OK
ftp.de.debian.org            36 ms    7 hops - OK
ftp.de.debian.org            37 ms    7 hops - OK
ftp.de.debian.org            38 ms    7 hops - OK


ftp.de.debian.org            36 ms   7 hops  100% ok (10/10) [   61]
   61 ftp.de.debian.org
#
```

Wie oben bereits angesprochen, erzeugt `netselect-apt` eine Datei `sources.list` im aktuellen Verzeichnis. Dazu verfügt es über einen weiteren Schalter, mit dem Sie die entsprechende Veröffentlichung (siehe Abschnitt 2.10) angeben und eine passende Liste dazu generieren lassen. `netselect-apt` akzeptiert dazu Angaben wie *stable* oder *unstable*, aber auch die Alternativnamen der Veröffentlichung wie *Wheezy* oder *Sid*.

Im nachfolgenden Beispiel kommt zusätzlich die Option `-o test.list` zum Einsatz, was dazu führt, dass `netselect-apt` die ermittelten Paketmirrors in diese Datei speichert. Ohne diese Angabe überschreibt es die aktuelle Paketliste unter `/etc/apt/sources.list`.

Speicherung der ermittelten Paketmirrors in einer separaten Datei

```
# netselect-apt stable -o test.list
Using distribution stable.
Retrieving the list of mirrors from www.debian.org...

--2014-02-13 14:55:02--  http://www.debian.org/mirror/mirrors_full
Auflösen des Hostnamen »www.debian.org (www.debian.org)«... 5.153.231.4, 130.89.148.14,  ↩
    2001:610:1908:b000::148:14, ...
Verbindungsaufbau zu www.debian.org (www.debian.org)|5.153.231.4|:80... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 338381 (330K) [text/html]
In »»/tmp/netselect-apt.WrCIoS«« speichern.

100%[=======================================================>] 338.381      959K/s    ↩
    in 0,3s

2014-02-13 14:55:03 (959 KB/s) - »»/tmp/netselect-apt.WrCIoS«« gespeichert [338381/338381]
```

```
Choosing a main Debian mirror using netselect.
netselect: 347 (23 active) nameserver request(s)...
Duplicate address 218.100.43.30 (http://ftp.au.debian.org/debian/, http://mirror.waia.asn. ↩
    au/debian/); keeping only under first name.
netselect: 343 (23 active) nameserver request(s)...
Duplicate address 195.222.33.229 (http://ftp.ba.debian.org/debian/, http://mirror.debian. ↩
    com.ba/debian/); keeping only under first name.
...
Running netselect to choose 10 out of 333 addresses.
...
The fastest 10 servers seem to be:

        http://artfiles.org/debian/
        http://ftp.plusline.de/debian/
        http://ftp5.gwdg.de/pub/linux/debian/debian/
        http://debian.netcologne.de/debian/
        http://ftp.uni-erlangen.de/debian/
        http://deb-mirror.de/debian/
        http://mirror.de.leaseweb.net/debian/
        http://mirror.1und1.de/debian/
        http://deb-mirror.de/debian/
        http://ftp.uni-bayreuth.de/debian/

Of the hosts tested we choose the fastest valid for HTTP:
        http://artfiles.org/debian/

Writing test.list.
Done.
#
```

Die von `netselect-apt` erzeugte Datei `test.list` enthält neben den Paketmirrors auch eine ganze Reihe Kommentare. Diese helfen Ihnen dabei, zu verstehen, wofür jeder einzelne Eintrag gedacht ist.

Inhalt der automatisch generierten Liste der Paketmirrors

```
# cat test.list

# Debian packages for stable
deb http://artfiles.org/debian/ stable main contrib
# Uncomment the deb-src line if you want 'apt-get source'
# to work with most packages.
# deb-src http://artfiles.org/debian/ stable main contrib

# Security updates for stable
deb http://security.debian.org/ stable/updates main contrib
#
```

Aus unserer Sicht lohnt sich der Aufruf von `netselect` bzw. `netselect-apt` bei stationären Systemen (Servern) mit fester Anbindung nur bedingt. Hilfreich ist das Vorgehen bspw. nach der ersten Einrichtung, einem Standortwechsel des Gerätes oder der Änderung der Infrastruktur, da letztere in der Regel häufig recht konstant ist. Bei Endsystemen an einem festen Ort raten wir Ihnen, die Werkzeuge nur interessehalber auszuprobieren, weil die Zugriffszeiten in diesem Kontext nicht immer eine so große Relevanz haben. Bei Systemen für die Infrastruktur wirkt sich die Optimierung hingegen meist weitaus stärker aus.

Bei mobilen Geräten sieht das hingegen deutlich anders aus. Mit Laptops oder Smartphones sind Sie variabler und den damit einhergehenden Schwankungen in der Netzanbindung stärker ausgesetzt. Auffällig wird die Anpassung dann, wenn Sie größere Entfernungen zurücklegen, bspw. ein Land oder einen Kontinent gewechselt haben.

3.5.2 Paketquellen mit `apt-spy` nach Bandbreite auswählen

In den meisten Fällen wissen Sie, welche Paketquellen Sie als Bezugspunkt für ihr Linuxsystem verwenden möchten, sei es aus Gewohnheit, langjähriger Erfahrung als Systembetreuer oder auf der Grundlage von Vorgaben für eine bereits bestehende

Infrastruktur. Sind Sie hingegen frei in ihrer Entscheidung, kann Ihnen das Programm `apt-spy` [Debian-Paket-apt-spy] helfen, die Paketquellen für eine netzbasierte Installation zu ermitteln, die von ihrem Standort aus am besten erreichbar sind.

Dazu untersucht `apt-spy` die Paketmirrors für die angefragte Region bzgl. ihrer Bandbreite, Erreichbarkeit und damit indirekt zur Verlustrate der übertragenen Netzwerkpakete. Ohne Belang ist dabei, wie weit der Paketmirror netztechnisch von ihrem Standort entfernt ist. Das kann auch dazu führen, dass schwächere, lokale Paketmirrors außen vor bleiben, wenn ein anderer, erreichbarer Paketmirror schneller und mit einer höheren Bandbreite verfügbar ist. Um einen Paketmirror einschätzen zu können, lädt `apt-spy` pro Spiegelserver etwa 10 MB an Daten herunter. Da die Liste der verfügbaren Paketmirrors mehr als einhundert Einträge umfasst, können Sie damit durchaus an das Limit ihrer Transferrate stoßen (wie bei uns im Test geschehen).

Im Ergebnis verringert sich die Bezugszeit von Paketen insgesamt. Insbesondere bei der Aktualisierung von Paketen (siehe Abschnitt 8.39) und im Rahmen eines Distributionsupgrades (siehe Abschnitt 8.45) ist das deutlich spürbar.

`apt-spy` erzeugt eine spezifische Liste der Paketquellen in der Datei `/etc/apt/sources.list.d/apt-spy.list`. Den Ausgangspunkt dafür bildet die aktuelle Liste der Paketmirrors von ftp.debian.org. Jeder darin enthaltene Eintrag wird von `apt-spy` auf dessen Erreichbarkeit von ihrem aktuellen Standort aus überprüft. Die vielversprechendsten Einträge landen in o.g. Liste.

Zu `apt-spy` gehört eine **Konfigurationsdatei** unter `/etc/apt-spy.conf`. Diese enthält die Länderkennungen sowie die Zuordnung von Länderkennungen und Kontinenten. Nachfolgender Ausschnitt zeigt die Bereiche für Nordamerika und Ozeanien.

Konfiguration von apt-spy für Nordamerika und Ozeanien (Ausschnitt)

```
$ cat /etc/apt-spy.conf
...
North-America:

CA
US
MX

Oceania:

AU
NZ
...
$
```

`apt-spy` versteht eine ganze Reihe von **Optionen zur Konfiguration**, von denen wir Ihnen eine Auswahl vorstellen. Weitere Optionen entnehmen Sie bitte der Manpage zum Programm.

`-d distribution`
: Debian-Veröffentlichung (Abschnitt 2.10), für die die Paketliste erstellt werden soll

`-a area`
: geographische Region, aus der die Paketmirrors kommen sollen

`-e Anzahl`
: *early finish* — legt die Anzahl der Paketmirrors fest, die Sie abfragen möchten

`-p proxy`
: Proxyserver in der Schreibweise `Hostname:Port`

`-s Land`
: eine Komma-separierte Liste von Länderkürzeln, aus der die Paketmirrors kommen sollen

`update`
: Aktualisieren der Liste der Paketmirrors

Nun folgen eine Reihe von **Beispielen aus der Praxis**. Beispiel 1 zeigt Ihnen den schnellsten Debian-Paketmirror in Europa für die Veröffentlichung Debian *stable*:

Ermittlung der Debian-Paketmirrors für die Region Europa

```
# apt-spy -d stable -a Europe

SERVER: ftp.at.debian.org
Benchmarking FTP...
                Downloaded 9001094 bytes in 3.22 seconds
                Download speed: 2731.88 kB/sec

SERVER: debian.sil.at
Benchmarking FTP...
                Downloaded 9001094 bytes in 4.19 seconds
                Download speed: 2096.46 kB/sec

SERVER: debian.sil.at
Benchmarking FTP...
                Downloaded 9001094 bytes in 3.66 seconds
                Download speed: 2404.31 kB/sec

SERVER: gd.tuwien.ac.at
Benchmarking HTTP...
                Downloaded 9001094 bytes in 2.31 seconds
                Download speed: 3810.42 kB/sec
...
Writing new sources.list file: /etc/apt/sources.list.d/apt-spy.list
#
```

In Beispiel 2 sehen Sie, wie sie ähnliches für Deutschland ermitteln:

Ermittlung der Debian-Paketmirrors für die Region Deutschland

```
# apt-spy -d stable -s de

SERVER: ftp.de.debian.org
Benchmarking FTP...
                Downloaded 8990025 bytes in 2.75 seconds
                Download speed: 3190.72 kB/sec

SERVER: debian.inf.tu-dresden.de
Benchmarking FTP...
                Downloaded 8990025 bytes in 3.32 seconds
                Download speed: 2643.16 kB/sec

...
Writing new sources.list file: /etc/apt/sources.list.d/apt-spy.list
#
```

Die dazu von `apt-spy` erzeugte Datei unter `/etc/apt/sources.list.d/apt-spy.list` sieht bspw. so aus:

Erzeugte Datei sources.list für Deutschland

```
# sources.list generated by apt-spy v3.2.2
#
# Generated using:
#
# apt-spy \
#       -d stable \
#       -s de
#
deb ftp://ftp.uni-erlangen.de/debian/ stable main #contrib non-free
deb-src ftp://ftp.uni-erlangen.de/debian/ stable main #contrib non-free
deb http://security.debian.org/ stable/updates main
```

Bitte beachten Sie, dass die ermittelten Paketmirrors stets von ihrem aktuellen Standort und der dort vorliegenden Netzwerkanbindung abhängen. Führen Sie das gleiche Kommando auf ihrem Linuxsystem aus, können die Ergebnisse von der Ausgabe im Buch abweichen.

3.6 Automatisiertes Auswählen von Paketquellen

3.6.1 DNS Round Robin

In den meisten Fällen gibt es zu einem Servernamen genau eine IP-Adresse (oder je eine IPv4- und eine IPv6-Adresse). Stärker in Anspruch genommene Dienste verteilen die Last aber oftmals auf mehr als eine Maschine. In solchen Fällen werden gerne mehr als eine IP-Adresse pro Servernamen zurückgegeben. Daraufhin wählt das Programm, welches eine Verbindung aufbauen möchte, willkürlich eine der zur Auswahl stehenden IP-Adressen aus. Auf diese Weise kann die Last zwischen mehreren (identisch konfigurierten) Servern aufgeteilt werden.

Ein bekanntes Beispiel eines solchen Falles im Kontext von Debian Paketspiegeln ist `ftp.us.debian.org`. Aufgrund der Größe der USA wird die Last des dortigen primären Debian-Paketmirrors auf drei Server aufgeteilt.

IP-Verteilung des primären Debian-Paketmirrors für die USA

```
$ host ftp.us.debian.org
ftp.us.debian.org has address 64.50.236.52
ftp.us.debian.org has address 128.61.240.89
ftp.us.debian.org has address 64.50.233.100
ftp.us.debian.org has IPv6 address 2610:148:1f10:3::89
$
```

Auch wenn es vier IP-Adressen sind, handelt es sich jedoch nur um drei Server. Sowohl die IPv6-Adresse `2610:148:1f10:3::89`, als auch die IPv4-Adresse `128.61.240.89` gehören zum Server `debian.gtisc.gatech.edu`.

3.6.2 Paketquellen über GeoIP auswählen

Bei diesem Verfahren wird einer IP-Adresse ein geographischer Standort zugeordnet. Die Genauigkeit dieser Funktion schwankt je nach Internet-Anbieter, Datenbank und eingesetztem Verfahren.

Zu beachten ist dabei, dass der angegebene Standort nicht notwendigerweise dem tatsächlichen Standort des Rechners mit dieser IP-Adresse entspricht. Meistens ist das der Standort des Providers, von dem Sie diese IP-Adresse bezogen haben bzw. der diese IP-Adresse vergeben hat.

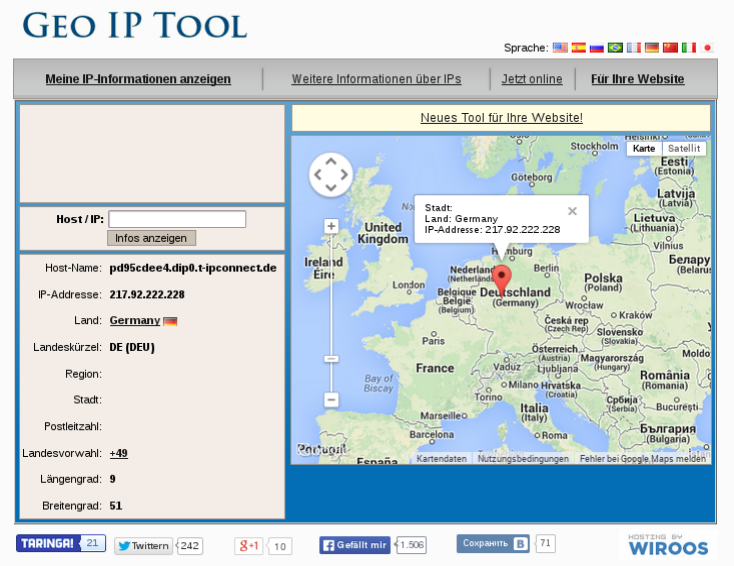

Abbildung 3.5: Standortlokalisierung über das GeoIP Tool [geoiptool]

Eine Offline-Zuordnung ermöglicht beispielsweise das Paket *geoip-database* [Debian-Paket-geoip-database]. Es enthält eine entsprechende Datenbank mit stets bestehenden, festen IP-Adressen. Darüberhinaus existieren jedoch auch deutlich exaktere Alternativen.

Diese Funktionalitat lässt sich nutzen, um anhand der anfragenden IP-Adresse automatisiert einen geographisch nahen Paketmirror zu finden. Bei Debian ist dies an mehreren Stellen im Einsatz.

3.6.3 Immer per GeoIP: Security-Updates

Security-Updates kommen bei Debian nicht über das normale Spiegelnetzwerk, welches regulär nur alle sechs Stunden aktualisiert wird. Stattdessen besteht ein separates Spiegelnetzwerk unter dem Hostnamen `security.debian.org`, das nur nach Bedarf aktualisiert wird. Dieses Spiegelnetzwerk verwendet immer GeoIP per DNS und gibt stets eine Liste von Paketspiegeln auf demselben Kontinent zurück, auf dem Sie sich netztechnisch gerade aufhalten.

3.6.4 GeoIP per DNS

Mit dem Hostnamen `cdn.debian.net`[3] erreichen Sie seit vielen Jahren (fast) immer einen offiziellen Debian-Spiegelserver in ihrer Nähe. `cdn.debian.net` war daher für einige Jahre die Voreinstellung im Debian-Installer, wenn Sie nicht explizit einen anderen Paketmirror ausgewählt haben.

Die Nähe wird bei diesem System über die Quell-IP der DNS-Anfrage bestimmt und hängt somit auch davon ab, wo der zwischenspeichernde DNS-Server steht, an den die DNS-Anfrage gesendet wurde. Netzwerktechnisch betrachtet stehen diese DNS-Server im Normalfall immer unweit des anfragenden Rechners.

Da hier die Auswahl des Paketmirrors bereits auf DNS-Ebene passiert, funktioniert diese Methode nicht nur mit HTTP, sondern auch mit FTP und theoretisch sogar auch mit `rsync`. Allerdings bieten nicht alle Paketmirror den Inhalt auch per `rsync` an. Somit funktioniert `cdn.debian.net` stets zuverlässig mit HTTP als auch mit FTP in der `/etc/apt/sources.list`. Nachfolgendes Beispiel zeigt Eintragungen für Debian *Wheezy* mit HTTP wie auch FTP:

```
deb http://cdn.debian.net/debian/ wheezy main contrib non-free
deb ftp://cdn.debian.net/debian/ wheezy main contrib non-free
```

[3] *CDN* steht für Content Distribution Network

Eine tatsächliche Nähe zum zurückgegebenen Paketmirror ist jedoch nicht immer gegeben. Zeitweise funktioniert dieser Dienst in manchen Ländern auch weniger gut. Nachfolgend sehen Sie einige Beispiele für ausgewählte Regionen. Das erste Beispiel stammt aus Deutschland:

Erreichbarkeit des Paketmirrors cdn.debian.net von Deutschland aus

```
$ host cdn.debian.net
cdn.debian.net is an alias for deb.cdn.araki.net.
deb.cdn.araki.net has address 137.226.34.42
deb.cdn.araki.net has address 129.143.116.10
deb.cdn.araki.net has address 195.71.68.86
$
```

In Frankreich bekommen Sie ggf. die folgende Ausgabe:

Erreichbarkeit des Paketmirrors cdn.debian.net von Frankreich aus

```
$ host cdn.debian.net
cdn.debian.net is an alias for deb.cdn.araki.net.
deb.cdn.araki.net has address 91.121.124.139
$
```

Ein Beispiel aus Schweden:

Erreichbarkeit des Paketmirrors cdn.debian.net von Schweden aus

```
$ host cdn.debian.net
cdn.debian.net is an alias for deb.cdn.araki.net.
deb.cdn.araki.net has address 213.129.232.18
$
```

Machen Sie die Anfrage hingegen aus Großbritannien, kann es so aussehen:

Erreichbarkeit des Paketmirrors cdn.debian.net von Großbritannien aus

```
$ host cdn.debian.net
cdn.debian.net is an alias for deb.cdn.araki.net.
deb.cdn.araki.net has address 83.142.228.128
$
```

Für die Schweiz sah es zum Zeitpunkt unserer Recherche leider etwas enttäuschend aus. Der Dienst war nicht verfügbar — wie man sieht, funktionert der Dienst eben nicht überall perfekt.

Erreichbarkeit des Paketmirrors cdn.debian.net von der Schweiz aus

```
$ host cdn.debian.net
Host cdn.debian.net not found: 3(NXDOMAIN)
$
```

3.6.5 GeoIP per HTTP-Redirect

Raphael Geissert hat den Debian Redirector [Debian-Redirector] entwickelt, der eine Alternative zu `cdn.debian.net` darstellt. Ursprünglich lief dieser Dienst unter der URL `http://http.debian.net/` außerhalb der Debian-internen Infrastruktur. Seit Mai 2015 ist der Dienst auf Debian-eigene Server umgezogen und dort unter `http://httpredir.debian.org/` zu erreichen. Die ursprüngliche Adresse `http://http.debian.net/` leitet seitdem automatisch auf `http://httpredir.debian.org/` um. Tragen Sie als Paketquelle `http://httpredir.debian.org/debian/` ein, sendet APT zuerst eine HTTP-Anfrage an `httpredir.debian.org` und bekommt als Antwort eine Weiterleitung an den eigentlichen Paketmirror. Die nachfolgende Ausschnitt zeigt den Eintrag für Debian 7 *Wheezy* in der Datei `/etc/apt/sources.list`:

```
deb http://httpredir.debian.org/debian/ wheezy main contrib non-free
```

Interessant ist auch die Netzwerkkommunikation, die (unbemerkt) im Hintergrund abläuft. Wir zeigen das anhand eines Beispiels aus der Schweiz genauer. Zur Analyse kommt das Kommando `GET` aus dem Paket *libwww-perl* zum Einsatz. Zu sehen ist, dass die Anfrage an `httpredir.debian.org` aufgelöst wird und eine Weiterleitung (Redirect) zum Debian-Paketmirror an der ETH Zürich erfolgt.

Auswertung der Netzwerkkommunikation bei der Auflösung der IP

```
$ GET -SUsed http://httpredir.debian.org/debian/
GET http://httpredir.debian.org/debian/
User-Agent: lwp-request/6.03 libwww-perl/6.04

302 Found
Connection: close
Date: Thu, 10 Jul 2014 00:32:59 GMT
Location: http://debian.ethz.ch/debian/
Vary: Accept-Encoding
Content-Length: 0
Content-Type: text/plain
Client-Date: Thu, 10 Jul 2014 00:32:59 GMT
Client-Peer: 46.4.205.44:80
Client-Response-Num: 1

GET http://debian.ethz.ch/debian/
User-Agent: lwp-request/6.03 libwww-perl/6.04

200 OK
Connection: close
Date: Thu, 10 Jul 2014 00:32:59 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Type: text/html;charset=UTF-8
Client-Date: Thu, 10 Jul 2014 00:32:59 GMT
Client-Peer: 129.132.53.171:80
Client-Response-Num: 1
Client-Transfer-Encoding: chunked
Title: Index of /debian
$
```

Von **Vorteil** ist hier die höhere Genauigkeit. Die GeoIP kann nicht nur auf den zwischenspeichernden DNS-Servern, sondern auch auf den anfragenden Rechner selbst angewendet werden. Dabei wird auch das genutzte Netzwerkprotokoll berücksichtigt. Nutzen Sie IPv6, erhalten Sie dann eine Empfehlung für einen passenden, IPv6-fähigen Paketmirror in ihrer Nähe [Debian-Mirror-Doku].

Desweiteren kann der Redirect auch in Abhängigkeit der angefragten Datei passieren. So werden z.B. Anfragen nach Paketen aus dem Bereich *Backports* nur an Paketmirrors weitergeleitet, die auch die Paketquellen für *Backports* spiegeln[4]. Darüber hinaus muss die Paketquelle nicht auf jeden Paketspiegel unter dem gleichen Pfad liegen. Möglich sind z.B. statt der Empfehlung `/debian/` auch `/pub/debian/` oder `/mirror/debian/`.

Das Verfahren mit HTTP-Weiterleitungen hat jedoch auch **Nachteile**. Einerseits funktioniert es ausschließlich per HTTP (oder HTTPS), da FTP keine Weiterleitungen kennt. Aus dieser Einschränkung leitet sich auch der Hostname `httpredir.debian.org` ab. Andererseits werden pro Paketliste sowie pro heruntergeladenem Paket stets zwei HTTP-Anfragen gesendet.

Da sich dieses Verfahren trotz der o.g. Einschränkung in der Praxis als zuverlässiger, flexibler, genauer und leichter wartbar erwies[5], setzt es sich gegenüber dem Dienst `cdn.debian.net` und somit innerhalb von Debian immer mehr als Voreinstellung durch.

3.6.6 Automatische Paketmirror-Auswahl per Mirror-Liste

APT kann seit Version 0.8 (ca. Ende 2010, ab Debian 6 *Squeeze* und Ubuntu 10.10 *Maverick Meerkat*) über das Schlüsselwort `mirror` in der Datei `/etc/apt/sources.list` seine Paketquelle aus einer Liste von Paketspiegeln aussuchen [Vogt-Apt-

[4] Dies ist nur noch für Debian 6 *Squeeze* relevant. Ab Debian 7 *Wheezy* sind die Backports in den normalen Debian-Paketquellen enthalten.

[5] Es ist wesentlich leichter installierbar als ein autoritativer DNS-Server für eine bestimmte Zone und der Quellcode ist per Git verfügbar.

Mirror].

Offizielle Mirror-Listen im passenden Format gibt es bisher jedoch nur von Ubuntu. Für Ubuntu 12.04 LTS *Precise Pangolin* sieht der Eintrag für generell gut erreichbare Paketmirrors wie folgt aus:

```
deb mirror://mirrors.ubuntu.com/mirrors.txt precise main restricted universe multiverse
```

In diesem Fall wird z.B. beim Aufruf von `apt-get update` zunächst die Mirror-Liste unter `http://mirrors.ubuntu.com/mirrors.txt` heruntergeladen. In dieser Datei stehen die Basis-URLs mehrerer Paketquellen. Danach sucht sich APT per Zufall eine der dieser Paketquellen aus und lädt von dort die spezifizierten Paketlisten herunter.

Clientseitig nutzt dieses Verfahren keinerlei GeoIP-Informationen, sondern wählt pro Maschine einen zufälligen Paketspiegel aus. Zunächst deutet o.g. URL auf eine simple Textdatei hin. Diese Datei wird jedoch bei jedem Aufruf automatisch neu generiert und — ähnlich wie die Weiterleitungen beim Debian Redirector — je nach anfragender IP dynamisch mit URLs anderer Spiegel gefüllt. Laden Sie diese Datei aus der Schweiz herunter, kann sie z.B. so aussehen:

```
http://ubuntu.ethz.ch/ubuntu/
http://archive.ubuntu.csg.uzh.ch/ubuntu/
http://mirror.switch.ch/ftp/mirror/ubuntu/
http://archive.ubuntu.com/ubuntu/
```

Aus Österreich sieht die Liste dagegen z.B. so aus:

```
http://ubuntu.lagis.at/ubuntu/
http://ubuntu.inode.at/ubuntu/
http://ubuntu.uni-klu.ac.at/ubuntu/
http://gd.tuwien.ac.at/opsys/linux/ubuntu/archive/
http://archive.ubuntu.com/ubuntu/
```

Erfragen Sie die Liste in Deutschland oder Frankreich, kommen sogar noch deutlich mehr Paketspiegel zur Auswahl. Eine Abfrage von einem Server, der bei dem deutschen Internetdienstleister Hetzner gehostet wird, ergab 34 aufgelistete Paketspiegel[6].

Auffällig ist allerdings, dass als letzter Paketmirror in dieser Liste jeweils immer auch noch `archive.ubuntu.com` angegeben wird. Unter diesem Hostnamen sind per DNS Round Robin wiederum zur Zeit sechs verschiedene Server von Canonical erreichbar.

Alternativ zum dynamisch generierten `mirrors.txt` können Sie bei Ubuntu auch eine Paketspiegel-Liste per Land angeben. Für Deutschland gibt es eine Liste von deutschen Ubuntu-Paketspiegeln unter `http://mirrors.ubuntu.com/DE.txt`. Diese verwenden Sie z.B. für Ubuntu 14.04 LTS *Trusty Tahr* wie folgt in der `/etc/apt/sources.list`:

```
deb mirror://mirrors.ubuntu.com/DE.txt trusty main restricted universe multiverse
```

Wenn Sie möchten, können Sie dieses Feature von APT natürlich auch nutzen, um eine Liste ihrer favorisierten Paketspiegel selbst zusammenzustellen — auch unter Debian.

Unter `http://www.debian-paketmanagement.de/hetzner-mirrors.txt` haben wir z.B. eine Liste von Paketspiegeln für Debian erstellt, die alle bei dem deutschen Internetdienstleister Hetzner gehostet sind (ohne Gewähr) und somit für andere ebenfalls dort gehostete Server nicht mit ins Trafficvolumen zählen. Der passende Eintrag in der `/etc/apt/sources.list` sind dann so aus:

```
deb mirror://www.debian-paketmanagement.de/hetzner-mirrors.txt wheezy main contrib non-free
```

3.6.7 Welcher Paketmirror wird schlussendlich benutzt?

Egal, ob Sie eine der o.g. Methoden zur automatischen Auswahl des Paketspiegels verwendet haben oder ob Sie einen bestimmten Hostnamen in ihrer `/etc/apt/sources.list` eingetragen haben — oft stellt sich die Frage: Von welchem Paketspiegel bezieht APT denn nun die Paketlisten und Pakete tatsächlich? APT gibt diese Information leider nicht allzu leicht preis.

[6] Um keine unübersichtlich langen Beispiele abzudrucken, wurden hier absichtlich die beiden Beispiele aus dem deutschsprachigen Raum gewählt, die relativ kurze Listen ergeben.

Falls einem der schlussendlich verwendeten Hostnamen mehr als eine IP zugewiesen ist, wird eine davon zufällig ausgewählt. APT und `aptitude` verwenden diese IP-Adresse intern, zeigen sie aber erst dann an, wenn Sie eines der Programme zur Paketverwaltung benutzen und die zusätzliche Option `-o Debug::pkgAcquire::Worker=true` verwenden. Damit wird APT sehr gesprächig und zeigt en detail, welche Einstellungen es benutzt. In dem nachfolgendem Beispiel sehen Sie das auszugsweise bei der Installation des Pakets *netselect-apt*.

Informationen zum tatsächlich genutzten Paketmirror bei der Verwendung von apt-get

```
# apt-get -o Debug::pkgAcquire::Worker=true install netselect-apt
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  netselect
The following NEW packages will be installed:
  netselect netselect-apt
0 upgraded, 2 newly installed, 0 to remove and 4 not upgraded.
Starting method '/usr/lib/apt/methods/http'
...
  -> http:600%20URI%20Acquire%0aURI:%20http://ftp.ch.debian.org/debian/pool/main/n/ ↩
      netselect/netselect_0.3.ds1-25_amd64.deb%0aFilename:%20/var/cache/apt/archives/partial ↩
      /netselect_0.3.ds1-25_amd64.deb%0a%0a
  <- http:102%20Status%0aURI:%20http://ftp.ch.debian.org/debian/pool/main/n/netselect/ ↩
      netselect_0.3.ds1-25_amd64.deb%0aMessage:%20Connecting%20to%20ftp.ch.debian.org ↩
      %20(129.132.53.171)
...
#
```

Deutlich übersichtlicher ist jedoch die Demo-Seite des Debian Redirectors [Debian-Redirector]. Neben dem aktuellen Standort — hier Berlin — zeigt Abbildung 3.6 die ausgewählten Paketquellen als Hostname an.

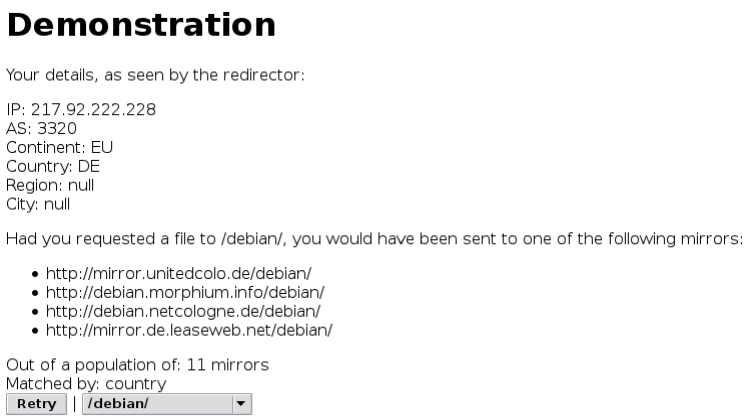

Abbildung 3.6: Auswahl des Paketmirrors über den Debian Redirector [Debian-Redirector]

Weitere Ansatzpunkte zur Leistungsfähigkeit eines bestimmten Mirrors liefern Ihnen die Werkzeuge `netselect` bzw. `netselect-apt`. Beide Programme stellen wir unter Bandbreite zum Paketmirror testen in Abschnitt 3.5 ausführlich vor.

3.7 `apt-setup` — Erstellung der Paketliste während der Installation

ToDo: Abschnitt veraltet?

Bei der Erst- oder Neuinstallation des Debian-Systems stellt der Debian Installer eine `/etc/apt/sources.list` zusammen, da diese ja bis dato noch nicht existiert. Bei der textbasierten Installation kommt das Programm `apt-setup` [Debian-Paket-apt-setup] zum Einsatz. Die Auswahl und Konfiguration der Paketquellen erfolgt dabei über diese schlichte Ncurses-Oberfläche.

Einschränkung zur Verwendung
Verwenden Sie dieses Programm nicht auf einem bereits installierten System — es ist nur für den Debian Installer gedacht. Daher beinhaltet der Paketname auch das Suffix `-udeb` (siehe Übergangs- und Metapakete in Abschnitt 2.7.2).

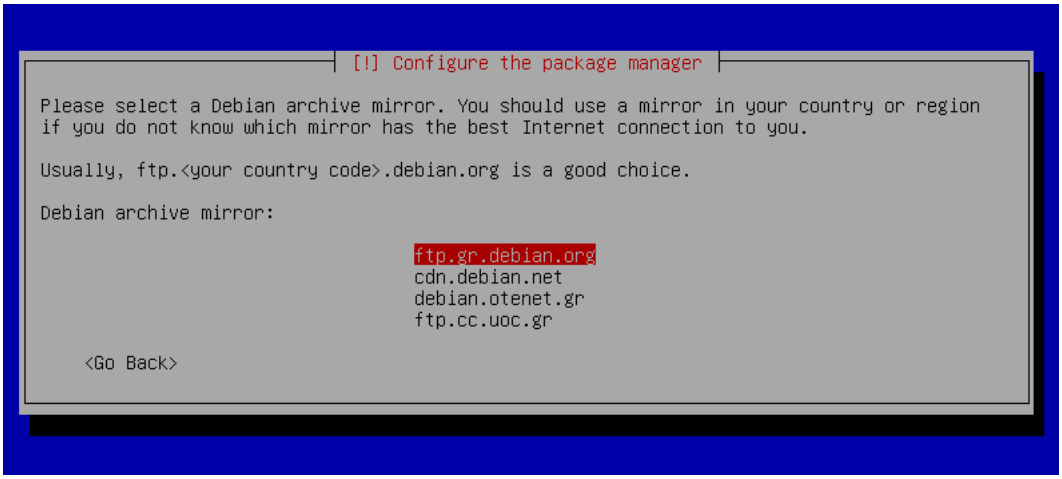

Abbildung 3.7: Auswahl der Paketquellen über `apt-setup`

3.8 Physische Installationsmedien mit `apt-cdrom` einbinden

Nutzen Sie keine netzbasierte Installation, sondern greifen auf die altbewährte Form anfassbarer Installationsmedien zurück, ist in diesem Fall das Programm `apt-cdrom` aus dem Paket *apt* die richtige Wahl (das Paket mit dem ähnlichen Namen *apt-cdrom-setup* [Debian-Paket-apt-cdrom-setup] ist lediglich für den Debian Installer vorgesehen). Damit fügen Sie ein bereitstehendes Installationsmedium zu Ihrer Liste der Paketquellen in der Datei `/etc/apt/sources.list` (siehe Abschnitt 3.3) hinzu.

Während vor wenigen Jahren noch eine CD gebräuchlich war, ist es heute aufgrund der Datenmenge eher eine DVD, eine Blu-ray oder ein entsprechendes Abbild eines Datenträgers, kurz genannt ISO-Image. `apt-cdrom` kann mit allen diesen Medien und Formaten umgehen.

Die Alternative zu `apt-cdrom` ist, alle neuen Medien von Hand nachzutragen. Das kann ein wenig aufwendig werden. `apt-cdrom` erleichtert Ihnen die Arbeit und übernimmt folgende Schritte:

- Medium (CD, DVD, Blu-ray, ISO-Image) auf Vollständigkeit und dessen Struktur überprüfen
- Validierung der Indexdateien des Mediums

Voraussetzung dafür ist jedoch, dass sich das Medium bereits im Laufwerk befindet, das Medium eingehängt ist und das dazugehörige Gerät (DVD-Laufwerk, etc.) einen passenden Mountpoint in der Datei `/etc/fstab` hat.

`apt-cdrom` unterstützt die folgenden nützlichen Aufufe:

`apt-cdrom`
kurze Information (Hilfeseite) zu `apt-cdrom`

`apt-cdrom add`
Installationsmedium hinzufügen

`apt-cdrom -d Verzeichnis add` (Langform `--cdrom`)
Installationsmedium aus dem angegebenen Verzeichnis hinzufügen, bspw. von einem USB-Stick

`apt-cdrom ident`
Identität des Installationsmediums ausgeben (siehe nachfolgende Beispielausgabe)

`apt-cdrom -r` (Langform `--rename`)
Umbenennen eines Mediums. Ändert den Namen eines Mediums oder überschreibt den Namen, der dem Medium gegeben wurde.

Identifikation eines Installationsmediums

```
# apt-cdrom ident
Verwendeter CD-ROM-Einbindungspunkt: /media/cdrom/
CD-ROM wird eingebunden.
Identifizieren ... [3e81e0fb1b74074c6e427e18afef3ab7-2]
Gespeicherte Kennzeichnung:
Einbindung der CD-ROM wird gelöst ...
#
```

3.9 Einträge mit `add-apt-repository` im Griff behalten

`add-apt-repository` ist ein Python-Skript, um Einträge automatisiert und damit leichter zur Liste der Paketquellen (siehe Abschnitt 3.3) hinzuzufügen oder auch wieder auszutragen. Es öffnet dazu die bestehende Liste der Paketquellen, ergänzt bzw. korrigiert die Einträge und überprüft diese zusätzlich auf Echtheit (siehe Abschnitt 3.12). Fehlende GPG-Schlüssel trägt es dabei automatisch in ihrem lokalen Schlüsselring nach. Indem es die vielen Einzelschritte kombiniert, spart es Zeit und lässt sich darüber hinaus auch problemlos zur Automatisierung in ihre eigenen Skripte integrieren.

`add-apt-repository` ist bis Debian 7 *Wheezy* Bestandteil des Pakets *python-software-properties* [Debian-Paket-python-software-properties] und ab Debian 8 *Jessie* in *software-properties-common* [Debian-Paket-software-properties-common]. Es stellt graphische Komponenten bereit, die bspw. auch im Rahmen von Synaptic (siehe Abschnitt 6.4.1) und dem Ubuntu Software Center (siehe Abschnitt 6.4.4) zum Einsatz kommen. Diese graphischen Komponenten beschreiben wir ausführlich unter Einstellungen mit Synaptic und im Ubuntu Software Center in Abschnitt 3.10.

Um die Handhabung auf der Kommandozeile noch weiter zu vereinfachen und insbesondere die Vertauschung der beiden Begriffe *apt* und *add* abzufangen, existiert zusätzlich das Kommando `apt-add-repository`. Dies ist durch einen symbolischen Link auf `add-apt-repository` realisiert.

3.9.1 Aufruf und Optionen

`add-apt-repository` akzeptiert als Parameter neben der Angabe des Repositories in Form einer vollständigen Zeile in korrekter Quotierung ebenso Personal Package Archives (PPAs) aus dem Ubuntu Launchpad [Ubuntu-Launchpad]. Der Aufruf ist von der Abfolge her analog zum manuellen Eintrag in der Liste der Paketquellen (siehe Abschnitt 3.3):

```
add-apt-repository deb uri distribution [component1] [component2] [...]
```

3.9.2 Beispiele

Möchten Sie das Repository namens *Petra* zu ihrer Installation von Linux Mint hinzufügen, funktioniert der folgende Aufruf:

```
add-apt-repository 'deb http://packages.linuxmint.com/ petra main'
```

Ein PPA-Archiv namens *gnome-desktop* für Ubuntu fügen Sie wie folgt hinzu:

```
add-apt-repository ppa:gnome-desktop
```

Um ein Repository wieder auszutragen, rufen Sie `add-apt-repository` mit dem zusätzlichen Schalter `--remove` auf. Nachfolgendes Beispiel zeigt das für den Eintrag für Medibuntu, aus dem der Zweig *non-free* wieder entfernt wird:

```
add-apt-repository --remove 'https://packages.medibuntu.org non-free'
```

3.10 Einstellungen mit Synaptic und im Ubuntu Software Center

Auch mittels Synaptic (siehe Abschnitt 6.4.1) und dem Ubuntu Software Center (siehe Abschnitt 6.4.4) können Sie die Datei `/etc/apt/sources.list` anpassen. Dazu öffnen Sie den entsprechenden Dialog unter dem Menüpunkt Einstellungen → Paketquellen. Unter Gnome/GTK erfolgt daraufhin ein Aufruf des Programms `software-properties-gtk` aus dem bereits weiter oben genannten Projekt Python Software Properties [Debian-Paket-python-software-properties]. Das Pendant unter KDE heißt `software-properties-kde`.

Über die verschiedenen Reiter stellen Sie die gewünschten Paketquellen ein. Abbildung 3.8 zeigt die Einstellungen zu den Standard-Debian-Repositories, Abbildung 3.9 zu weiteren Paketquellen anhand des Ubuntu Software Center.

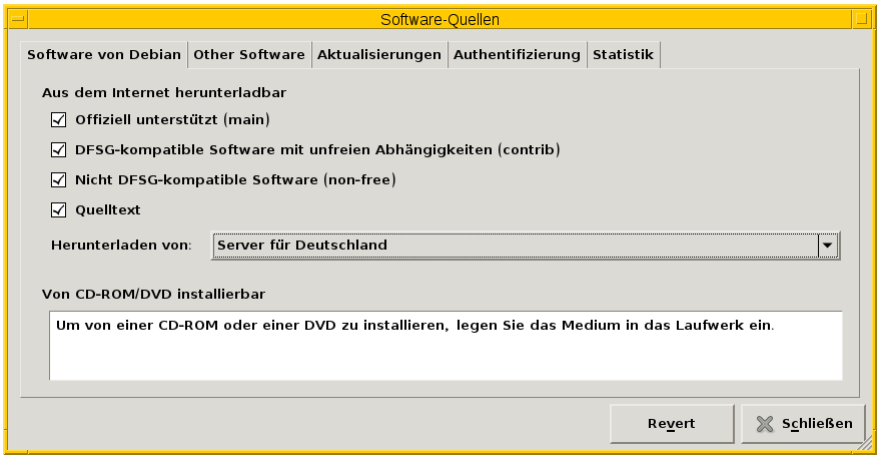

Abbildung 3.8: Einstellung der Komponenten in Synaptic

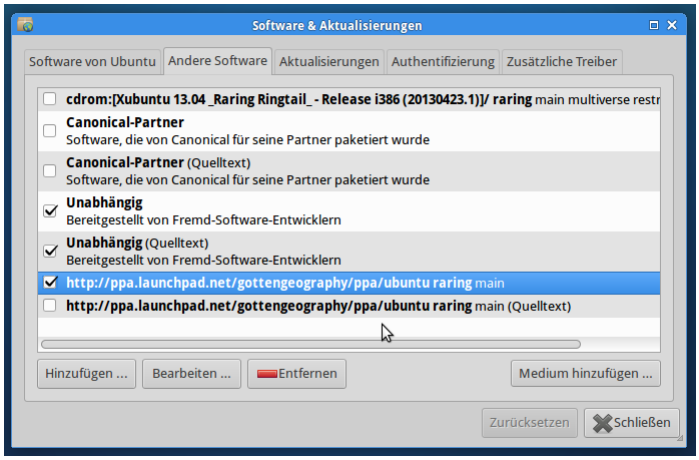

Abbildung 3.9: Festlegung der Paketquellen für andere Software im Ubuntu Software Center

3.11 Debian und Ubuntu Sources List Generator

Möchten Sie ihre Liste der Paketquellen nicht von Hand zusammentragen, sondern stattdessen über eine Software erstellen lassen, bieten sich der Debian Sources List Generator [Debian-Sources-List-Generator] und der Ubuntu Sources List Generator [Ubuntu-Sources-List-Generator] von Jonhnatha Trigueiro an. Dabei handelt es sich um eine JavaScript-Anwendung, die Sie über ihren Webbrowser benutzen. Als Ergebnis erhalten Sie am Ende eine Textdatei, die Sie überprüfen und in Ihr System einpflegen können. Vor der Übernahme empfehlen wir, für alle Fälle von der derzeit genutzten Datei eine Sicherheitskopie anzulegen, sodass Sie im Bedarfsfall darauf zurückgreifen können.

3.11.1 Feinheiten für Debian

Zunächst wählen Sie ihre geographische Region aus, danach die Veröffentlichung (siehe Abschnitt 2.10), die Architektur (siehe Abschnitt 1.2) Ihres Systems und die Distributionsbereiche (siehe Abschnitt 2.9). In der rechten Spalte wählen Sie die gewünschten Repositories von Drittanbietern aus, sofern Sie dazu Bedarf haben (siehe Abbildung 3.10). Über den Knopf Generate sources list erstellt Ihnen das Programm eine passende Liste der Paketquellen (siehe Abbildung 3.11). Diese Datei können Sie nun als neue `/etc/apt/sources.list` in Ihr System übernehmen.

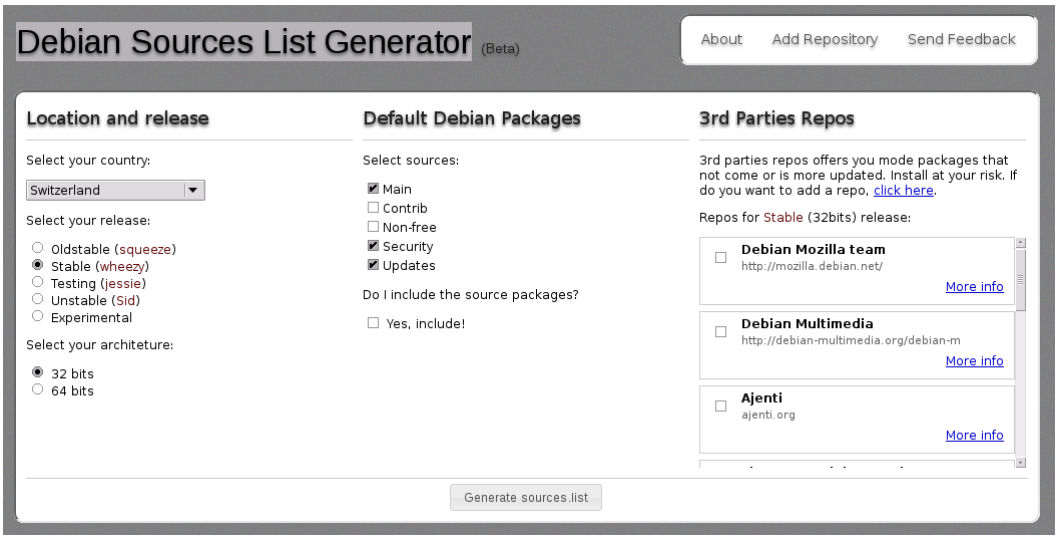

Abbildung 3.10: Auswahl der Komponenten im Debian Sources List Generator

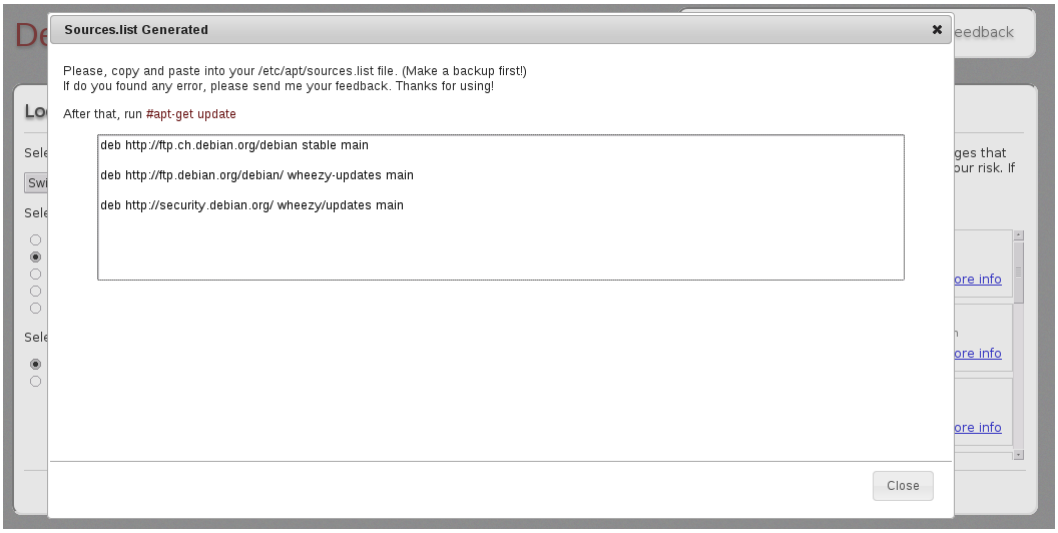

Abbildung 3.11: Erzeugte `sources.list` durch den Debian Sources List Generator

3.11.2 Feinheiten für Ubuntu

Die Abfolge ist ähnlich zu Debian, nur wesentlich umfangreicher. Nach der geographischen Region und der Veröffentlichung (siehe Abschnitt 2.10) wählen Sie die Distributionsbereiche (siehe Abschnitt 2.9) aus, die hier als Ubuntu Branches bezeichnet werden. Hinter den Fragezeichen verbergen sich Erläuterungen, welche den ausgewählten Distributionsbereich näher beschreiben. Danach können Sie neben der Architektur (siehe Abschnitt 1.2) auch etliche zusätzliche Paketquellen von Ubuntu-Partnern

hinzufügen. Am Schluß erstellen Sie mit einem Klick auf den Knopf Generate die entsprechende Liste der ausgewählten Paketquellen, die Sie in ihr System übernehmen können.

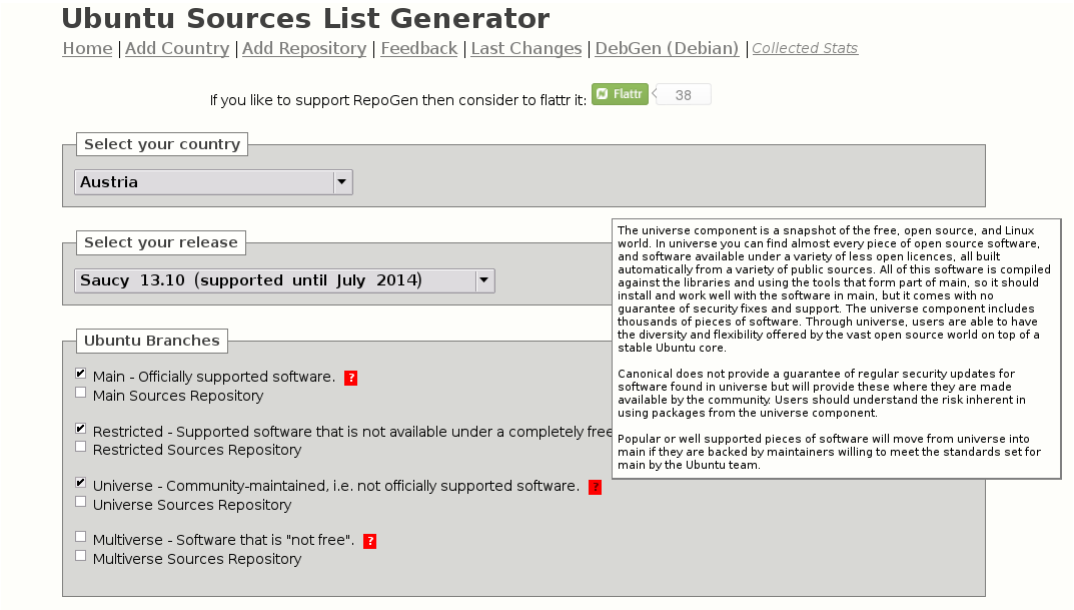

Abbildung 3.12: Auswahl der Komponenten im Ubuntu Sources List Generator

3.12 Paketquelle auf Echtheit überprüfen

3.12.1 Basiswissen

Paketquellen und Repositories sind im Prinzip Fileserver mit einer vorab festgelegten, spezifischen Struktur, deren Inhalt öffentlich zugänglich ist [Debian-Wiki-Debian-Repository-Format]. Vereinfacht betrachtet muss bei dessen Abruf sichergestellt werden, dass die von dort bezogenen Daten echt sind und auch mit den Originaldaten übereinstimmen, aus denen die Distribution besteht. Daher sind in der Paketverwaltung mehrstufige Mechanismen integriert, welche die Echtheit und Vollständigkeit der empfangenen Paketlisten und Pakete überprüfen (Authentizität).

Hintergrund dafür ist einerseits, dass eine Paketquelle Paketarchive unterschiedlichster Herkunft umfasst. Die Daten könnten aus einer wenig vertrauenswürdigen Quelle stammen und auch Schadcode enthalten. Ebenso nimmt die Zuverlässigkeit von Speichermedien (Datenträger) mit der Zeit ab und sorgt für fehlerhafte Bitfolgen. Desweiteren erfolgt der Transport über Leitungsnetze unterschiedlichster Art, wobei hier gekippte Bits und somit Übertragungsfehler und verfälschte Daten auf dem Transportweg nicht vollständig auszuschließen sind.

Daher sind sowohl alle Veröffentlichungen (siehe Abschnitt 2.10), als auch die Paketquellen (siehe Abschnitt 3.1) mit den Paketlisten und die darüber bereitgestellten, einzelnen Pakete jeweils separat digital signiert. Eine digitale Signatur („Schlüssel“, GPG-Key) besteht aus zwei Teilen — einem öffentlichen und einem privaten, geheimen Schlüssel. Die Paketlisten werden zunächst vom Verwalter des Repositorys mit seinem privaten Schlüssel signiert und der dazugehörige, öffentliche Schlüssel bekanntgegeben bzw. als Paket hinterlegt. Mit Hilfe dieses Signatur-Paares überprüfen Sie einerseits die Echtheit der Paketquelle und andererseits über die Hashsummen jeden einzelnen Pakets in den Paketlisten auch jedes einzelne Paket daraus (siehe auch „Bezogenes Paket verifizieren“ in Abschnitt 8.35).

APT und `aptitude` haben diesen Vorgang in ihre internen Abläufe integriert und nehmen Ihnen diesen Verifizierungsschritt vollständig ab. Falls die Signatur korrekt ist, dann wird der Paketmirror bzw. das bezogene Paket als glaubwürdig eingeschätzt. Falls nicht, erhalten Sie eine deutliche Warnung.

3.12.2 Schlüsselverwaltung mit `apt-key` (Überblick)

Die Verwaltung der Schlüssel erfolgt mit dem Programm `apt-key`. Dazu gehört ein Schlüsselring mit allen GPG-Schlüsseln der Paketquellen, aus denen Pakete bezogen wurden. Bei Debian sind diese Schlüssel Bestandteil des Pakets *debian-archive-keyring* [Debian-Paket-debian-archive-keyring], bei Ubuntu heißt das Paket hingegen *ubuntu-keyring* [Ubuntu-Paket-ubuntu-keyring].

Der primäre Schlüsselring für lokale, als vertrauenswürdig eingestufte Schlüssel ist die Datei `/etc/apt/trusted.gpg`. Für zusätzliche Schlüsselbunde und Dateifragmente weiterer vertrauenswürdiger Schlüssel ist das Verzeichnis `/etc/apt/trusted.gpg.d/` vorgesehen. Insbesondere o.g. Schlüsselbund-Pakete speichern ihre Schlüsselbund-Dateien in diesem Verzeichnis.

Die einzelnen Dateien in `/etc/apt/trusted.gpg.d/` gelten als Konfigurationsdateien, können also vom lokalen Administrator verändert oder gelöscht werden. Deswegen sind diese Schlüssel zusätzlich auch noch in der Datei `/usr/share/keyrings/debian-archive-keyring.gpg` gespeichert.

Die Schlüssel haben eine begrenzte Gültigkeit oder können auch zurückgezogen werden. Daher sind in der Schlüsselbund-Datei `/usr/share/keyrings/debian-archive-removed-keys.gpg` auch noch die abgelaufenen oder zurückgezogenen Schlüssel vergangener Debian-Veröffentlichungen verfügbar.

Ähnliche Schlüsselringe gibt es auch für andere Veröffentlichungen, bspw. `debian-edu-archive-keyring` für Skolelinux/DebianEdu ([Skolelinux]) und `debian-ports-archive-keyring` für das Debian-Ports-Projekt (Abschnitt 1.2.1.

3.12.3 Unterkommandos von `apt-key`

Mit `apt-key` greifen Sie auf ihren gespeicherten Schlüsselring zu. Damit lassen Sie sich bspw. die gemerkten Schlüssel anzeigen, fügen neue Schlüssel zum Schlüsselring hinzu oder entfernen diese daraus wieder. Diese Vorgänge kommen meist dann zum tragen, wenn Sie Ihr Debian-System von Ballast befreien und nicht mehr benötigte Schlüssel austragen oder weitere Paketquellen einbinden möchten, deren Schlüssel (noch) nicht offiziell hinterlegt ist.

Die vier Unterkommandos `list`, `finger`, `export` und `exportall` haben rein informativen Charakter. Mit den ersten beiden zeigen sie zu den gespeicherten, vertrauenswürdigen Schlüsseln deren Erstell- und Verfallsdatum sowie den Eigentümer bzw. Aussteller des Schlüssels an. Im vorliegenden Fall ist dieser keine Person, sondern eine Debian-Veröffentlichung bzw. deren Verantwortlicher. Als E-Mail-Adresse ist hier diejenige der FTP-Master hinterlegt (siehe Abbildung 3.13).

Mit dem Aufruf `apt-key finger` zeigen Sie zusätzlich deren Fingerabdruck an (siehe Abbildung 3.14). Mit `apt-key export Schlüssel` geben Sie nur einen bestimmten Schlüssel auf der Standardausgabe als als PGP-Block aus. Der Schalter `apt-key exportall` führt das gleiche für alle Schlüssel durch.

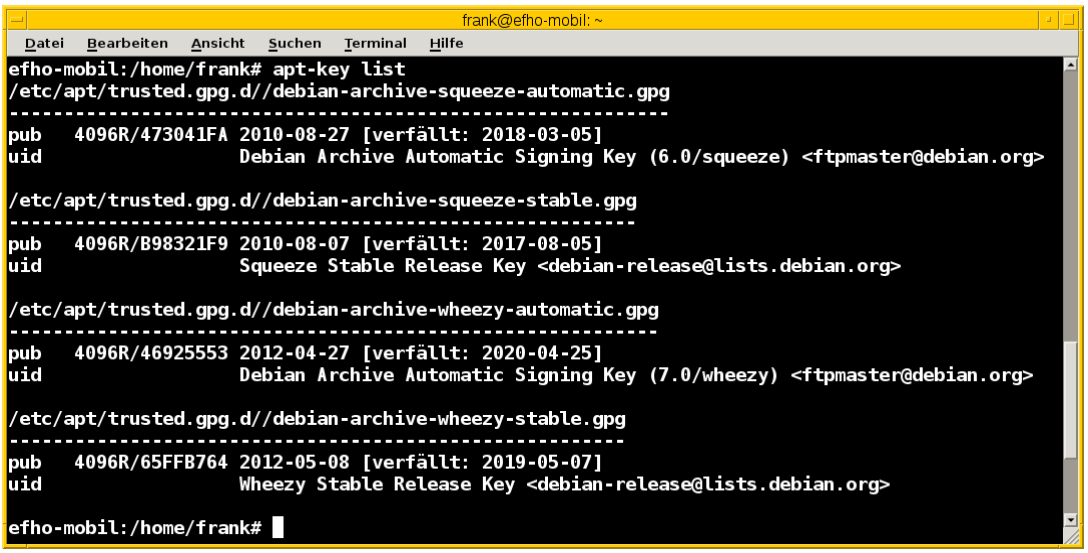

Abbildung 3.13: Auflistung der gespeicherten, vertrauenswürdigen Schlüssel

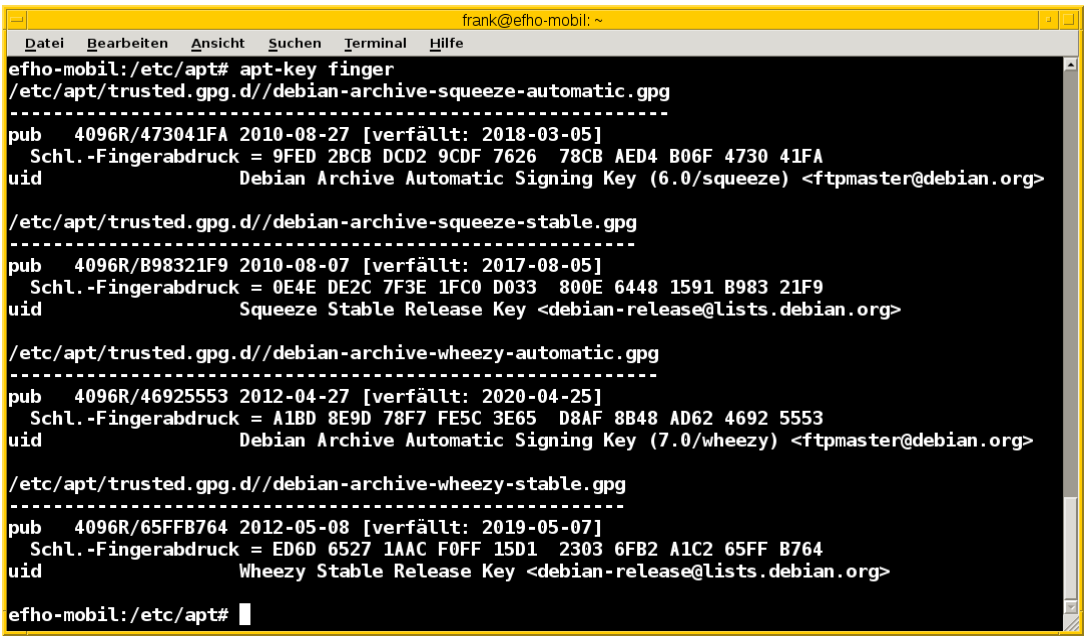

Abbildung 3.14: Darstellung der Fingerabdrücke der vertrauenswürdigen Schlüssel

Mit `apt-key add Schlüsseldatei` und `apt-key del Schlüssel-ID` verändern Sie den Inhalt des Schlüsselbundes. Mit ersterem fügen Sie einen neuen Schlüssel aus einer Datei hinzu, mit letzterem löschen Sie den Schlüssel mit der angegebenen Schlüssel-ID aus dem Schlüsselring.

Die Option `update` synchronisiert hingegen den lokalen Schlüsselbund mit dem Archivschlüsselbund. Dabei werden die Schlüssel aus dem lokalen Schlüsselbund entfernt, die nicht mehr gültig sind. In Ubuntu ist auch die Option `net-update` anwendbar, die eine Synchronisation mit einem Schlüsselbund über das Netzwerk ermöglicht.

3.12.4 Beispiel: Ergänzung eines Schlüssels

Nutzen Sie beispielsweise den Webbrowser Opera, finden Sie dazu keine Pakete in den offiziellen Debian-Paketquellen. Opera ist nicht als freie Software eingeordnet, aber als `deb`-Paket von der Herstellerwebseite beziehbar. Daher fügen Sie in Schritt eins die Paketquelle zur Datei `/etc/apt/sources.list` hinzu (siehe auch Abschnitt 3.3):

```
deb http://deb.opera.com/opera stable non-free
```

Als Schritt zwei benötigen Sie noch den dazugehörigen Schlüssel der Paketquelle. Der Hersteller empfiehlt auf seiner Seite den Bezug mittels `wget` wie folgt:

Bezug des Schlüssels zur Paketquelle, hier für Opera mittels wget

```
# wget http://deb.opera.com/archive.key
--2014-06-17 23:54:43--  http://deb.opera.com/archive.key
Auflösen des Hostnamen »deb.opera.com (deb.opera.com)«... 185.26.183.130
Verbindungsaufbau zu deb.opera.com (deb.opera.com)|185.26.183.130|:80... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 2437 (2,4K) [application/pgp-keys]
In »»archive.key«« speichern.

100%[=====================================================================>] 2.437       ↩
    --.-K/s   in 0s

2014-06-17 23:54:43 (63,0 MB/s) - »»archive.key«« gespeichert [2437/2437]
#
```

Unverschlüsselte Übertragung von Schlüsseln
Bitte beachten Sie, dass dieser Schlüssel jedoch nicht über gesicherte Kanäle (z.B. per HTTPS) heruntergeladen wurde und Sie damit nicht hundertprozentig sicher sein können, dass dieser Schlüssel wirklich von Opera ist. Leider scheint der Schlüssel auch nicht mit allzuvielen Signaturen ausgestattet zu sein, sodass eine Verifizierung über die Signaturen ebenfalls nicht möglich ist.

Der bezogene Schlüssel befindet sich nun im aktuellen Verzeichnis in der lokalen Datei `archive.key`. Diesen Schlüssel fügen Sie nun über den Aufruf `apt-key add archive.key` Ihrem lokalen Schlüsselbund hinzu:

Hinzufügen des bezogenen Schlüssels mittels apt-key

```
# apt-key add archive.key
OK
#
```

Hat alles geklappt, meldet sich `apt-key` mit einem schlichten `OK` zurück. Von nun an werden alle Pakete von dieser Paketquelle als vertrauenswürdig eingestuft. Auch Aktualisierungen über APT und `aptitude` sind problemlos möglich.

Es bleibt jedoch ein unangenehmer Beigeschmack erhalten. Aufgrund der ungesicherten Übertragung des bezogenen Schlüssels können Sie nicht sicher sein, ob der bezogene Schlüssel wirklich von Opera ist und Sie ihm vertrauen können, oder ob nicht zufällig eine Man-in-the-Middle-Attacke im Gange ist.

3.12.5 Alternative Benutzerschnittstellen zur APT-Schlüsselverwaltung

Neben dem Kommandozeilenprogramm `apt-key` existieren auch noch zwei interaktive Bedienoberflächen dazu: das auf GTK aufbauende `gui-apt-key` aus dem gleichnamigen Paket [Debian-Paket-gui-apt-key] und das auf Ncurses aufbauende `curses-apt-key` [curses-apt-key]. Beide besprechen wir hier nur kurz.

`gui-apt-key` starten Sie zunächst als Benutzer `root` oder mittels `sudo`. Im Dialogfenster (Abbildung 3.15) sehen Sie die Inhaber und das Ablaufdatum aller von APT als vertrauenswürdig eingestuften GPG-Schlüssel. Über das Menü haben Sie die Möglichkeit, weitere Schlüssel aus Dateien zu importieren, die Schlüssel gegen den Debian-Archiv-Schlüsselring zu aktualisieren (analog zu `apt-key update`), einen Schlüssel aus der Liste zu löschen oder Details wie einen Fingerabdruck zu einem Schlüssel anzeigen zu lassen.

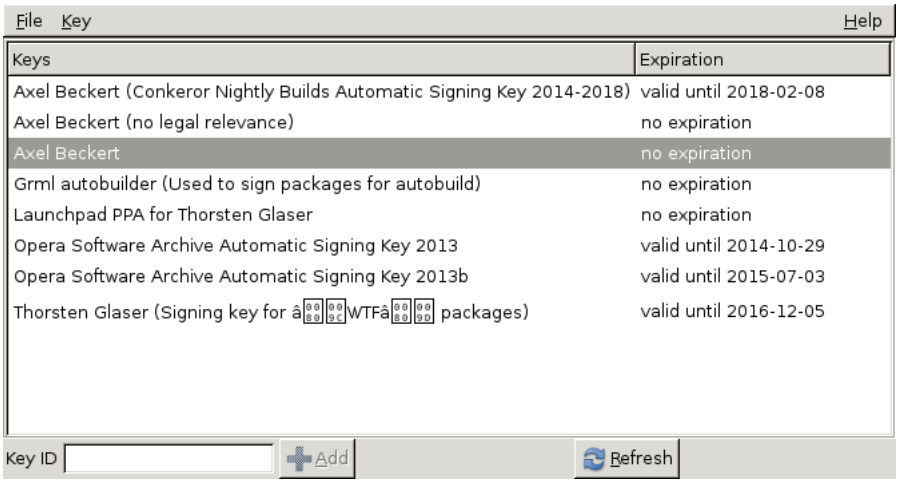

Abbildung 3.15: Hauptfenster von gui-apt-key

`curses-apt-key` nutzt dieselben Backend-Bibliotheken wie `gui-apt-key`. Daher bietet es die gleichen Funktionalitäten, braucht jedoch dazu keine graphische Umgebung und eignet sich daher insbesondere für die Nutzung auf Servern (siehe Abbildung 3.16).

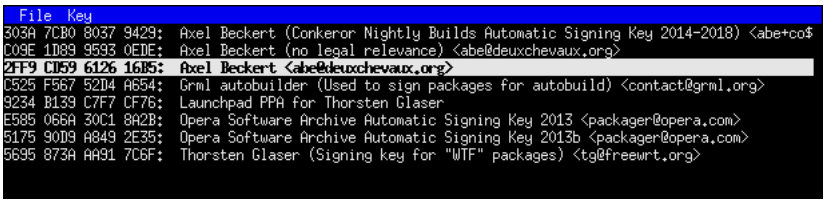

Abbildung 3.16: curses-apt-key in einem xterm

Derzeit ist `curses-apt-key` noch nicht Bestandteil von Debian und nur auf GitHub verfügbar [curses-apt-key]. Eine Aufnahme in Debian ist jedoch geplant [curses-apt-key-itp].

3.13 Liste der verfügbaren Pakete aktualisieren

Bevor Sie Veränderungen am Paketbestand veranlassen, empfehlen wir Ihnen, stets die Liste der lokal genutzten Pakete auf den neuesten Stand zu bringen. Damit arbeiten Sie mit den aktuellen Referenzen auf die bestehenden Softwarepakete. Diesen Schritt ermöglichen alle Werkzeuge zur Paketverwaltung.

Dazu bestehen verschiedene Möglichkeiten, die im Endeffekt alle das gleiche bewirken:

- Das klassische Kommando, das auch stets auf älteren Veröffentlichungen funktioniert, ist `apt-get update`. Auf neueren Veröffentlichungen, die das Kommando `apt` kennen, funktioniert auch `apt update` (siehe Abschnitt 6.2.2).
- `aptitude` (siehe Abschnitt 6.3.2) gestattet einen Aufruf über die Kommandozeile mittels `aptitude update`. Möchten Sie die Paketliste aktualisieren und danach interaktiv im Text-Modus weiterarbeiten, so rufen Sie `aptitude -u` auf. Sind Sie bereits im interaktiven Text-Modus von `aptitude`, sorgt der Tastendruck **u** für frische Paketlisten und die aktualisierte Darstellung in `aptitude`. Alternativ stoßen Sie die Aktion über den Menüeintrag `Aktionen -> Paketlisten aktualisieren` an.
- Bei Synaptic (siehe Abschnitt 6.4.1) verbirgt sich dieser Vorgang hinter dem Menüeintrag `Bearbeiten -> Paketinformationen neu laden`. Alternativ nutzen Sie dafür die Tastenkombination Ctrl-R.
- Im Programm SmartPM (siehe Abschnitt 6.4.3) lösen Sie die Aktualisierung für alle Paketquellen über den Menüpunkt `File -> Update channels` aus. Möchten Sie nur eine einzige Paketquelle auf den neuesten Stand bringen, wählen Sie stattdessen zunächst `File -> Update selected channels ...` aus und entscheiden danach, welche Paketquelle Ihres Erachtens eine Auffrischung verdient hat (siehe dazu Abbildung 3.17).

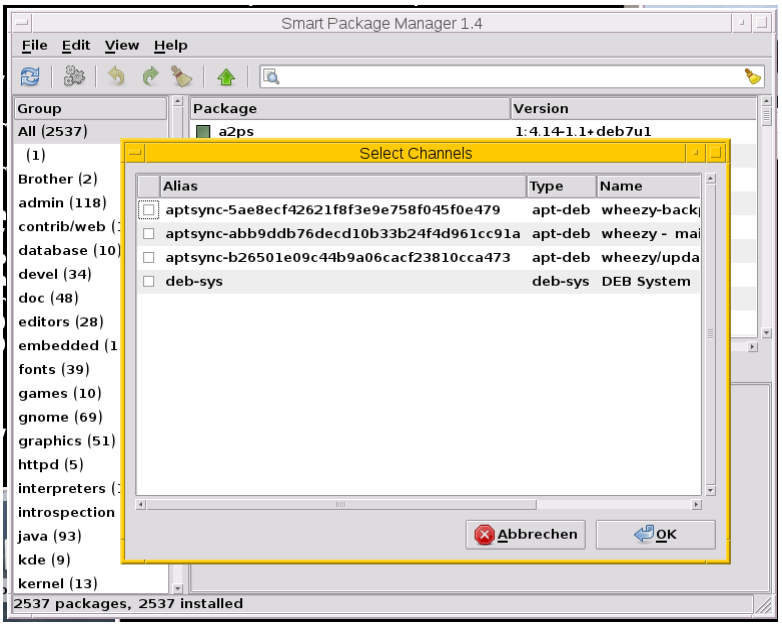

Abbildung 3.17: Auflistung der verfügbaren Paketquellen in SmartPM

Aktualisierung mit dpkg

`dpkg` kennt auch die beiden Schalter `--update-avail` und `--merge-avail`. Es setzt dafür lokal bereitstehende Paketlisten voraus, mit denen es dann seine Paketdatenbank unter `/var/lib/dpkg/available` aktualisiert. Diesen Schalter von `dpkg` betrachten wir hier nicht näher.

Führen Sie eines der o.g. Aufrufe aus, wird zunächst die Liste der Paketquellen in `/etc/apt/sources.list` (siehe Abschnitt 3.3) gelesen. Jeder Eintrag darin bezeichnet eine Paketquelle. Von diesen Paketquellen wird nacheinander jeweils eine aktuelle Liste der Pakete bezogen, die von dieser angegebenen Paketquelle verfügbar sind.

Jede bezogene Liste wird danach auf deren Echtheit geprüft. Dazu ist diese digital signiert (siehe Abschnitt 3.12). Mit Hilfe des GPG-Schlüssels für die Paketquelle prüfen `apt-get` bzw. `aptitude` automatisch deren Authentizität und falls ohne Beanstandung, vereinigen sie die bezogene Liste mit der bereits bestehenden, lokalen Paketliste (siehe Abschnitt 3.14). Dabei geben insbesondere `apt-get` und `aptitude` eine Reihe von Mitteilungen auf dem Terminal aus. Diese bedeuten:

- `Holen:1 Bezugsquelle Release.gpg`: beziehe den GPG-Schlüssel zur Veröffentlichung (siehe Abschnitt 2.10) von der als URL angegebenen Paketquelle (siehe Abschnitt 3.12)
- `OK Bezugsquelle [Datenmenge]`: der GPG-Schlüssel ist in Ordnung, die Signatur stimmt (siehe auch Abschnitt 3.12)
- `Holen:2 Bezugsquelle [Datenmenge]`: beziehe die Paketliste von der unter 1 als URL angegebenen Paketquelle
- `Ign Bezugsquelle`: Ein beim Herunterladen aufgetretener Fehler wird ignoriert (z.B. fehlende Übersetzungen)

Am Ende der Ausgabe erfolgt noch eine Zusammenfassung, welche Datenmenge in welcher Zeitspanne bezogen wurde. Nachfolgend sehen Sie die Ausgabe am Beispiel von `apt-get update`:

Aktualisierung der Paketliste durch apt-get update

```
# apt-get update
OK   http://ftp.de.debian.org wheezy Release.gpg
Holen: 1 http://security.debian.org wheezy/updates Release.gpg [836 B]
Holen: 2 http://security.debian.org wheezy/updates Release [102 kB]
OK   http://ftp.de.debian.org wheezy Release
OK   http://ftp.de.debian.org wheezy/main Sources
```

```
Holen: 3 http://security.debian.org wheezy/updates/main Sources [79,2 kB]
OK   http://ftp.de.debian.org wheezy/contrib Sources
OK   http://ftp.de.debian.org wheezy/non-free Sources
OK   http://ftp.de.debian.org wheezy/main i386 Packages
Holen: 4 http://security.debian.org wheezy/updates/contrib Sources [14 B]
OK   http://ftp.de.debian.org wheezy/contrib i386 Packages
Holen: 5 http://security.debian.org wheezy/updates/non-free Sources [14 B]
OK   http://ftp.de.debian.org wheezy/non-free i386 Packages
Holen: 6 http://security.debian.org wheezy/updates/main i386 Packages [150 kB]
OK   http://ftp.de.debian.org wheezy/contrib Translation-en
OK   http://ftp.de.debian.org wheezy/main Translation-de_DE
OK   http://ftp.de.debian.org wheezy/main Translation-de
Holen: 7 http://security.debian.org wheezy/updates/contrib i386 Packages [14 B]
OK   http://ftp.de.debian.org wheezy/main Translation-en
Holen: 8 http://security.debian.org wheezy/updates/non-free i386 Packages [14 B]
OK   http://ftp.de.debian.org wheezy/non-free Translation-en
Holen: 9 http://security.debian.org wheezy/updates/contrib Translation-en [14 B]
Holen: 10 http://security.debian.org wheezy/updates/main Translation-en [88,7 kB]
Holen: 11 http://security.debian.org wheezy/updates/non-free Translation-en [14 B]
Es wurden 421 kB in 0 s geholt (428 kB/s).
Paketlisten werden gelesen... Fertig
#
```

Überprüfung der Paketsignaturen
Konnten bei der Aktualisierung für neue Paketlisten keine gültigen Signaturen gefunden werden, wird eine Warnung ausgegeben. Entsprechende Zeilen beginnen mit `W:`. Pakete, die nicht korrekt signiert sind, können Schadcode enthalten und sollten nicht installiert werden. Zur Überprüfung auf korrekte Pakete tragen Sie bitte den passenden GPG-Key nach. Näheres dazu finden Sie unter Bezogenes Paket verifizieren in Abschnitt 8.35.

Veröffentlichung wechseln
Möchten Sie neuere Versionen von Paketen installieren oder auf eine andere Veröffentlichung von Debian wechseln, ist zusätzlich ein *upgrade* bzw. *dist-upgrade* erforderlich. Weitere Informationen dazu erhalten Sie unter Pakete aktualisieren in Abschnitt 8.39 bzw. Distribution aktulisieren in Abschnitt 8.45.

3.14 Lokale Paketliste und Paketcache

Die Paketverwaltung — genauer APT — pflegt lokale Paketlisten im Verzeichnis `/var/lib/apt/lists`. Diese Paketlisten dienen als Nachschlagewerk für APT. Wollen Sie den Paketbestand auf Ihrem Debian-System ändern, benutzt APT diese Paketliste, um die Existenz, die Verfügbarkeit von einer Paketquelle und die Abhängigkeiten eines Pakets zu bestimmen, bevor diese tatsächlich bezogen werden. Installieren Sie ein Paket nach (Abschnitt 8.36), weiß APT aus der lokalen Paketliste, von welcher Paketquelle und unter welcher URL es dieses herunterladen kann.

Die hier verwendete mehrstufige Vorgehensweise hat ihren Ursprung in der Anfangszeit von Debian, bei der der Internetzugang und dessen (nahezu) permanenter Verfügbarkeit noch nicht so selbstverständlich wie heute waren. Lokal verfügbare Informationen waren (und sind) stets mit geringerer Verzögerung nutzbar als externe Ressourcen und reduzieren zudem die Netzlast.

Die nachfolgende Auflistung ist typisch, wenn Sie als Paketmirror `ftp.de.debian.org` und die Distributionsbereiche *main*, *contrib* und *non-free* benutzen:

Übersicht zu den lokalen Dateien, in denen die Paketlisten hinterlegt sind

```
$ ls /var/lib/apt/lists
ftp.de.debian.org_debian_dists_wheezy_contrib_source_Sources
ftp.de.debian.org_debian_dists_wheezy_main_source_Sources
ftp.de.debian.org_debian_dists_wheezy_non-free_source_Sources
```

```
ftp.de.debian.org_debian_dists_wheezy_Release
ftp.de.debian.org_debian_dists_wheezy_Release.gpg
lock
partial/
security.debian.org_dists_wheezy_updates_contrib_binary-i386_Packages
security.debian.org_dists_wheezy_updates_contrib_i18n_Translation-en
security.debian.org_dists_wheezy_updates_contrib_source_Sources
security.debian.org_dists_wheezy_updates_main_binary-i386_Packages
security.debian.org_dists_wheezy_updates_main_i18n_Translation-en
security.debian.org_dists_wheezy_updates_main_source_Sources
security.debian.org_dists_wheezy_updates_non-free_binary-i386_Packages
security.debian.org_dists_wheezy_updates_non-free_i18n_Translation-en
security.debian.org_dists_wheezy_updates_non-free_source_Sources
security.debian.org_dists_wheezy_updates_Release
security.debian.org_dists_wheezy_updates_Release.gpg
$
```

Für jede Paketquelle aus `/etc/apt/sources.list` wird eine eigene, lokale Datei gepflegt. Diese ist eine Textdatei und beinhaltet alle Informationen zu den beziehbaren Paketen, bspw. den genauen Paketnamen und dessen Version (Abschnitt 2.11), den Maintainer des Pakets, die Paketabhängigkeiten zum Bauen des Pakets, die genutzte Architektur (Abschnitt 1.2), das Format des Debianpakets sowie die Checksummen der Pakete und das Sourcepaket (Abschnitt 2.7), aus der das Paket entstanden ist. Danach folgen die Projektwebseite sowie das Verzeichnis, in dem das Paket auf dem Paketmirror abgelegt ist. Zum Schluss stehen die Priorität, der Distributionsbereich (Abschnitt 2.9) und die Paketkategorie (Abschnitt 2.8). Nachfolgender Kasten zeigt die Informationen anhand des Pakets *easyspice* aus der Paketkategorie Elektronik (*electronics*).

Paketinformationen zum Paket easyspice

```
Package: easyspice
Binary: easyspice
Version: 0.6.8-2
Maintainer: Gudjon I. Gudjonsson <gudjon@gudjon.org>
Build-Depends: debhelper (>= 5), autotools-dev, libgtk2.0-dev, quilt
Architecture: any
Standards-Version: 3.7.3
Format: 1.0
Files:
 2a6ef8e1bd3e38220d44754e107c55ca 1037 easyspice_0.6.8-2.dsc
 6ffaab8c2dcdfc30ecdca52f3c8bcded 201627 easyspice_0.6.8.orig.tar.gz
 e4b3a00ad900341424cb6052e9f56a53 2607 easyspice_0.6.8-2.diff.gz
Checksums-Sha1:
 d76bcb68824a1866598bc453c81af2e0c4a4366d 1037 easyspice_0.6.8-2.dsc
 37ae8dea4522254c3de25f300f6fc9b568b87c10 201627 easyspice_0.6.8.orig.tar.gz
 ec13581034d9ef441a2cd7acddc0f5289c5f6c57 2607 easyspice_0.6.8-2.diff.gz
Checksums-Sha256:
 5b119deafa50846c6736b8b63e0be8c18e407e18b31c17a43590e2301cc39bec 1037 easyspice_0.6.8-2. ↩
    dsc
 55158436e05e372d48e03e59edc2dc2b49fbf366629a2943b265eb4562f1e975 201627 easyspice_0.6.8. ↩
    orig.tar.gz
 7c5d3460457e89f69d050e42d394a89d95536b1efd178c99c49e505e7807f3fa 2607 easyspice_0.6.8-2. ↩
    diff.gz
Homepage: http://easy-spice.sourceforge.net
Directory: pool/contrib/e/easyspice
Priority: source
Section: contrib/electronics
```

TODO: Querverweis auf `cron-apt` und `/etc/cron.daily/apt`.

Die Paketlisten ändern sich, wenn Aktualisierungen sowie neue Versionen von Paketen verfügbar werden und die Paketquellen auf den Spiegelservern entsprechend aktualisiert wurden. Daher raten wir Ihnen, die lokalen Paketlisten in regelmäßigen Abständen ebenfalls zu aktualisieren, bspw. mit den Aufrufen `apt-get update`, `aptitude update` oder einem anderen Werkzeug zur Paketverwaltung (Kapitel 6). Wie das genau vorsichgeht, erklären wir unter Liste der verfügbaren Pakete aktualisieren in Abschnitt 3.13 genauer.

Sollte die Aktualisierung fehlschlagen, könnte sich die Paketliste in einem inkonsistenten Zustand befinden. Wie Sie mit dieser Situation umgehen, erklären wir Ihnen unter Lokale Paketliste reparieren in Abschnitt 3.15 genauer.

3.15 Lokale Paketliste reparieren

Es kann vorkommen, dass eine lokale Paketliste, die im Verzeichnis `/var/lib/apt/lists` liegt, bei deren Aktualisierung (siehe Abschnitt 3.13) kaputtgeht. Das kommt sehr selten vor, aber bspw. dann, wenn nicht mehr genügend freier Speicherplatz für die neue Paketliste zur Verfügung steht oder das Entpacken der komprimierten Liste fehlschlägt. Sie bekommen das mit, wenn APT jammert und in Folge seine Arbeit verweigert.

APT versucht von sich aus, eine defekte oder nicht mehr vorhandene Liste wieder zu reparieren. Dazu beauftragen Sie APT mit dem Kommando `apt-get update`. Es bezieht die aktuellen Paketlisten von den in `/etc/apt/sources.list` angebenen Paketquellen und ersetzt die bereits bestehenden lokalen Paketlisten. Falls diese nicht mehr vorhanden sein sollten, legt APT diese Listen neu an.

Ist das erfolglos, räumen Sie als nächsten Schritt das Verzeichnis `/var/lib/apt/lists/partial` auf. Darin hinterlegt APT alle Zwischenstände und Teillisten. Löschen Sie dazu in besagtem Verzeichnis sämtliche Dateien und wiederholen danach das Kommando `apt-get update`.

Wenn das noch nicht geholfen hat, bereinigen Sie auch das Verzeichnis `/var/lib/apt/lists`. Danach wiederholen Sie das Kommando `apt-get update`.

Sollte das Vorgehen immer noch nicht von Erfolg gekrönt sein, warten Sie bitte sieben Stunden und wiederholen danach das Kommando `apt-get update` erneut. Hintergrund für die Bitte um Geduld ist die Erneuerung der Debian-Paketquellen auf den Paketmirrors (siehe Abschnitt 3.4). Diese Erneuerung erfolgt automatisiert alle sechs Stunden und bereinigt auch eventuelle Inkonsistenzen der Paketlisten auf dem Paketmirror. Um auch sicherzugehen, dass die Liste der Paketquellen auf dem neuesten Stand ist, warten Sie lieber einen kleinen Moment länger.

3.15.1 Aktualität des Mirrors überprüfen

Sollten die Fehler trotz Ihrer intensiven Bemühungen bestehen bleiben, bleibt noch eine Überprüfung, ob der angefragte Spiegelserver auch aktuell ist. Eine Zustandsübersicht über alle offiziellen Spiegelserver, die bei Debian registriert sind, finden Sie unter dem Debian Mirror Checker [Debian-Mirror-Checker] (siehe Abbildung 3.18).

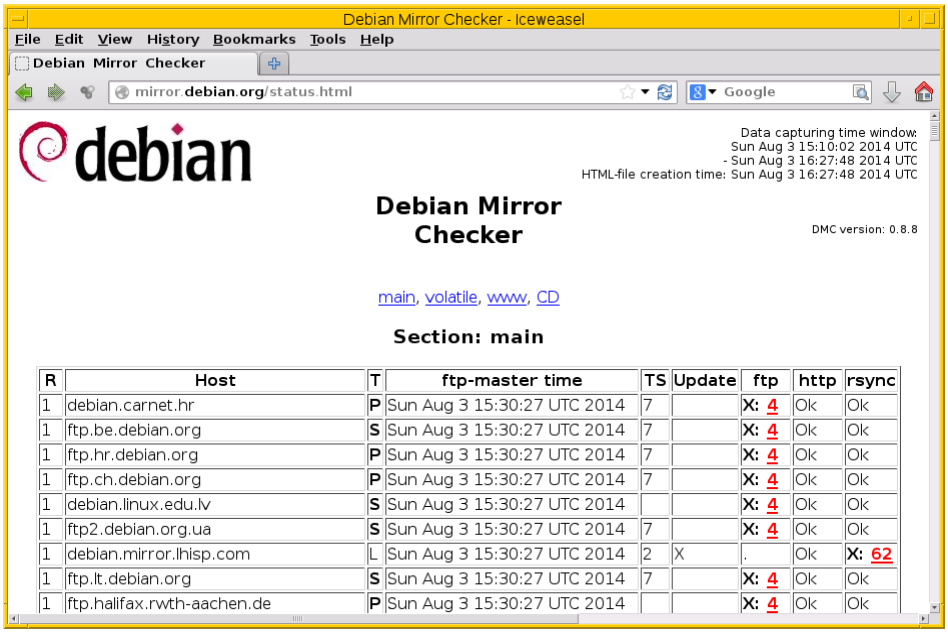

Abbildung 3.18: Status der verschiedenen Debian-Paketmirror

Wann die letzte Aktualisierung des von Ihnen gewählten Debian-Mirrors passiert ist, sehen Sie im Unterverzeichnis `project/trace/`, z.B. unter http://debian.ethz.ch/debian/project/trace/ für den Paketmirror der ETH Zürich. In dieser Liste suchen Sie nach dem Datumsstempel der Datei, die dem Hostnamen Ihres Spiegelservers entspricht. Wenn der Spiegelserver unter mehreren Namen erreichbar ist, finden Sie dort trotzdem nur einen davon. Es sollte immer die neuste Datei sein.

Wenn die gefundene Datei deutlich älter als sechs Stunden ist, prüfen Sie bitte zuerst, wann die letzte Aktualisierung des Mirror-Netzwerkes stattgefunden hat[7]. Unter [dinstall-status] finden Sie eine Datei, in der festgehalten wird, in welchem Schritt der Aktualisierung und Zustand sich der Master-Mirror gerade befindet. Ein Eintrag „all done“ bedeutet, dass zur Zeit keine Aktualisierung läuft. Das Endedatum zeigt den Zeitpunkt an, an dem die erste Stufe des Mirror-Netzwerkes mit den neuen Paketlisten und Paketen versorgt wurde.

Ob zur Zeit eine Aktualisierung Ihres gewählten Mirrors läuft, sehen sie an der Existenz einer Datei `Archive-Update-in-Progress` (ggf. erweitert um den bzw. einen Hostnamen des Spiegelservers) im Wurzelverzeichnis des APT-Repositorys, z.B. http://debian.ethz.ch/debian/Archive-Update-in-Progress-debian.ethz.ch.

[7] Bei Ausfällen oder Umbauten in der Infrastruktur wie auch kurz vor neuen Veröffentlichungen kann es durchaus vorkommen, dass der Abstand zwischen zwei Aktualisierungen des Mirrors deutlich mehr als sechs Stunden dauert, teilweise auch einen oder wenige Tage.

Kapitel 4

Debian-Paketformat im Detail

4.1 Konzepte und Ideen dahinter

Die Paketbeschreibung ist eine Textdatei[1]. Die Paketbeschreibung in den Paketen selbst erfolgt in englischer Sprache, wird aber für die Paketlisten auf den Spiegelservern von Debians Übersetzungsteams auch in andere Sprachen übersetzt.

Jedes Element der Beschreibung ist ein Schlüssel-Wert-Paar, wobei die Trennung zwischen Schlüssel und Wert durch einen Doppelpunkt erfolgt. Der *Schlüssel* ist ein aus der Umgangssprache abgeleiteter Begriff, der die Relation zwischen zwei oder mehr Paketen näher beschreibt. *Wert* ist hingegen eine Aufzählung von Paketen, die mit einem Komma voneinander getrennt werden. Ein ähnliches Konzept kommt bei den Kopfzeilen von E-Mails zum Tragen.

Zusätzlich kann ein Wert mit einer Aussage zu einer bestimmten Softwareversion ergänzt worden sein. Eine solche *versionierte Abhängigkeit* kann unterschiedliche Relationen umfassen. Tabelle 4.1 zeigt die derzeit zulässigen Operatoren samt einem Beispiel aus der Praxis.

Tabelle 4.1: Relationen für versionierte Abhängigkeiten

Operator	Beschreibung	Beispiel
`<<`	früher als	`xpdf-utils (<< 3.00)`
`<=`	früher oder gleich	`python-cairo (<=1.85)`
`=`	exakt gleich	`xfwm4 (=4.1)`
`>=`	gleich oder später	`libc6 (>=2.4)`
`>>`	später als	`libaa1 (>> 1.4)`

4.1.1 Binärpakete

Die folgenden Schlüsselworte werden in Binärpaketen (siehe Abschnitt 2.7.1) und den Paketlisten von diesen verwendet:

`Package`
: zu dt.: Paket; Name des Pakets ohne Versionsnummer und Architektur, siehe auch Benennung eines Debian-Pakets in Abschnitt 2.11

`Source`
: zu dt.: Quelle; Name des Quellpakets („source package“), aus dem das Binärpaket gebaut wurde, siehe auch Sourcepakete in Abschnitt 2.7.4

[1] früher teilweise im Encoding ISO 8859-1, heute nur noch in UTF-8

`Version`
zu dt.: Version oder Variante; Versionsnummer des Pakets, siehe Benennung eines Debian-Pakets in Abschnitt 2.11

`Architecture`
zu dt. Architektur oder Plattform; Basis, für die das Paket gebaut wurde oder *all*, falls das Paket architekturunabhängig ist, siehe Debian-Architekturen in Abschnitt 1.2

`Maintainer`
zu dt.: Betreuer, Verantwortlicher; Für das Paket verantwortliche Person oder Gruppe („Maintainer“ des Pakets) und dessen Erreichbarkeit als E-Mail-Adresse (siehe auch Paket nach Maintainer finden in Abschnitt 8.21)

`Homepage`
zu dt.: Internetpräsenz; Webseite des Projekts der paketierten Software oder Daten

`Installed-Size`
zu dt.: Installationsgröße; Speicherplatz, den das Paket auf dem Zielsystem belegen wird, nachdem es dort installiert wurde

`Depends`
zu dt.: hängt ab von; Name der installierten und konfigurierten Pakete und ggf. deren Versionsnummer, von dem das vorliegende Paket abhängt

`Pre-Depends`
zu dt.: hängt ab vorher von; Name der installierten und konfigurierten Pakete und ggf. deren Versionsnummer, von dem das vorliegende Paket und dessen Installationsskripte abhängen. Dies bedeutet, dass diese Abhängigkeiten vollständig installiert und ausgepackt sein müssen, bevor das Paket von `dpkg` ausgepackt werden darf.

`Recommends`
zu dt.: empfiehlt; Name der Pakete, welche als Ergänzung empfohlen werden und in den meisten Fällen ebenfalls gebraucht werden. Es ist ein Gegenstück zum Schlüsselwort `Enhances`.

`Suggests`
zu dt.: schlägt vor; Name der Pakete, welche als Ergänzung empfohlen werden. Es ist ein Gegenstück zum Schlüsselwort `Enhances`

`Conflicts`
zu dt.: kollidiert bzw. steht in Konflikt mit; Name der Pakete und ggf. deren Versionsnummer, mit denen es nicht gleichzeitig installiert sein darf

`Breaks`
zu dt.: bricht, verhindert, beschädigt; Name der Pakete und ggf. deren Versionsnummer, mit denen es nicht gleichzeitig verwendet werden kann

`Enhances`
zu dt.: erweitert, ergänzt, wertet auf; Benennt das Paket, welches es erweitert. Es ist das Gegenstück zu den Schlüsselworten `Suggests` und `Recommends`

`Replaces`
zu dt.: ersetzt; Name der Pakete, dessen Dateien es (teilweise) ersetzt

`Provides`
zu dt.: stellt bereit; Name der virtuellen Pakete, welche es bereitstellt, siehe Virtuelle Pakete in Abschnitt 2.7.5

`Section`
zu dt.: Sektion oder Paketkategorie, in die das Paket einsortiert ist, siehe Paketkategorien in Abschnitt 2.8

`Priority`
zu dt.: Priorität; Prioritätsstufe des Pakets, siehe Paket-Priorität und essentielle Pakete in Abschnitt 2.13

`Essential`
zu dt.: essentiell; Ihr Debian-System kann kaputt gehen, wenn dieses Paket entfernt wird, siehe dazu auch Markierung Essentiell in Abschnitt 2.13.6

Description
: zu dt.: Beschreibung; Dieses Feld enthält die Paketbeschreibung. Dabei ist die erste Zeile ein kurzer, einzeiliger Text und die darauf folgenden, eingerückten Zeilen beinhalten eine lange und ggf. über mehrere Absätze gehende, ausführlichere Beschreibung. Zwischen der Kurz- und Langbeschreibung kann auch ein Punkt (.) stehen.

Built-Using
: zu dt.: gebaut mit; Dieses Feld muss gemäß Debian Policy Manual §7.8 [Debian-Policy-Manual] vorhanden sein, sofern der Inhalt des Binärpakets nicht nur aus Quellcode aus dessen Quellpaket besteht und die Lizenz dieses Quellpakets vorschreibt, dass auch sämtlicher mit einkompilierter Quellcode frei verfügbar sein muss. Dies ist z.B. der Fall, wenn eine unter GNU GPL stehende Software statisch kompiliert wird oder Quellcode unter GNU GPL aus einem anderen Paket in das Binärpaket hineinkopiert wird (bspw. bei Stylesheets oder Hintergrundbildern für generierte Dokumentationen im HTML-Format).

Das nachfolgende Beispiel zeigt alle genutzten Elemente anhand der PDF-Bibliothek *poppler-utils*:

```
Package: poppler-utils
Source: poppler
Version: 0.18.4-6
Architecture: amd64
Maintainer: Loic Minier <lool@dooz.org>
Installed-Size: 445
Depends: libc6 (>= 2.4), libcairo2 (>= 1.10.0), libfreetype6 (>= 2.2.1), liblcms1 (>= ↩
    1.15-1), libpoppler19 (>= 0.18.4), libstdc++6 (>= 4.1.1)
Recommends: ghostscript
Conflicts: pdftohtml
Breaks: xpdf-utils (<< 3.02-2~)
Replaces: pdftohtml, xpdf-reader, xpdf-utils (<< 3.02-2~)
Provides: pdftohtml, xpdf-utils
Section: utils
Priority: optional
Multi-Arch: foreign
Homepage: http://poppler.freedesktop.org/
Description: PDF utilities (based on Poppler)
 Poppler is a PDF rendering library based on Xpdf PDF viewer.
 .
 This package contains command line utilities (based on Poppler) for getting
 information of PDF documents, convert them to other formats, or manipulate
 them:
  * pdffonts -- font analyzer
  * pdfimages -- image extractor
  * pdfinfo -- document information
  * pdfseparate -- page extraction tool
  * pdftocairo -- PDF to PNG/JPEG/PDF/PS/EPS/SVG converter using Cairo
  * pdftohtml -- PDF to HTML converter
  * pdftoppm -- PDF to PPM/PNG/JPEG image converter
  * pdftops -- PDF to PostScript (PS) converter
  * pdftotext -- text extraction
  * pdfunite -- document merging tool
```

4.1.2 Sourcepakete

In Sourcepaketen (siehe Abschnitt 2.7.4) sind neben den weiter oben genannten Schlüsselworten auch die folgenden Einträge zulässig:

Source
: zu dt.: Quelle; Name des Quellpakets.

Binary
: zu dt.: Binärdatei; Liste aller Binärpakete, die aus diesem Quellpaket gebaut werden.

`Package-List`
zu dt. Paketliste; Auflistung aller Binärpakete, die aus diesem Quellpaket gebaut werden. Zusätzlich werden das Paketformat (`deb` oder `udeb`), die Paketkategorie („Sektion“), die Priorität und die Architektur benannt.

`Format`
zu dt.: Format; verwendetes Format des Quellpakets, z.B. `1.0`, `3.0 (quilt)` oder `3.0 (native)` (siehe Aufbau und Format in Abschnitt 4.2).

`Architecture`
zu dt. Architektur oder Plattform; Im Gegensatz zu den Binärpaketen sind hier mehr als nur eine einzige Architektur zulässig. Es beinhaltet alle Architekturen, auf denen das Paket gebaut werden kann. Der Wert *any* bedeutet, dass das Paket auf jeder Architektur gebaut werden kann und soll (siehe Abschnitt 1.2).

`Uploaders`
zu dt.: Hochlader; bezeichnet die Liste der Co-Maintainer und Beitragenden des Pakets.

`Standards-Version`
zu dt.: Version der Standardisierung; Angabe, welcher Version des Debian Policy Manuals [Debian-Policy-Manual] dieses Paket entspricht.

`Vcs-Git`, `Vcs-Svn`, `Vcs-Hg`, `Vcs-Cvs`, `Vcs-Mtn`
zu dt.: Versionskontrollsystem; Angabe, von wo Sie eine aktuelle Entwicklungskopie des Quellpakets aus einem Versionskontrollsystems auschecken können.

`Vcs-Browser`
zu dt.: Versionskontrollsystem und Webbrowser; URL einer Webansicht des unter *Vcs-Git* u.a. genannten Repositories des Versionskontrollsystems.

`Build-Depends`
zu dt.: Abhängigkeiten beim Bauen von Paketen; Pakete, die notwendig sind, um alle architektur-abhängigen Binärpakete aus diesem Quellpaket zu bauen, sowie um das Build-Verzeichnis zu säubern („clean“-Ziel). Pakete, die als „essential“ (unbedingt notwendig) oder „build-essential“ (für den Bau von Paketen unbedingt notwendig) markiert sind, müssen nicht aufgelistet werden (Kommt fast immer vor.)

`Build-Depends-Indep`
zu dt.: Abhängigkeiten beim Bauen von Paketen (architekturunabhängig); Pakete, die zusätzlich zu den unter `Build-Depends` aufgelisteten Paketen notwendig sind, um auch die architektur-unabhängigen Pakete aus diesem Quellpaket zu bauen. Hier sind meist die Pakete aufgelistet, die notwendig sind, um die Dokumentation oder Übersetzungsdateien zu bauen. (Kommt meist nur bei komplexeren Quellpaketen vor.)

`Build-Conflicts`
zu dt. Bau-Konflikte; Pakete, die nicht installiert sein dürfen, wenn die architektur-abhängigen Binärpakete aus diesem Quellpaket gebaut werden sollen. Dies sind meistens Pakete, die das `configure`-Skript beim Testen der notwendigen Bibliotheken stören oder aber Pakete, die zusätzliche, unerwünschte Abhängigkeiten in den gebauten Binärpaketen verursachen würden. (Kommt selten vor.)

`Build-Conflicts-Indep`
zu dt. Bau-Konflikte (architekturunabhängig); Pakete, die nicht installiert sein dürfen, wenn die architektur-unabhängigen Binärpakete aus diesem Quellpaket gebaut werden sollen. (Kommt sehr selten vor.)

`Files`, `Checksums-Sha1`, `Checksums-Sha256`
MD5-, SHA1- und SHA256-Checksummen sowie Dateinamen und -größen der enthaltenen Quellcode-Archive.

`Testsuite`
Optionales Feld, das angibt, mit welchem Programm das installierte Paket auf Funktionalität getestet werden kann. Derzeit ist der einzige mögliche Wert `autopkgtest` (siehe Debian Enhancement Proposal *DEP 8* [DEP-8] und das gleichnamige Debianpaket dazu [Debian-Paket-autopkgtest].

4.1.3 Weitere Metadaten

In den Paketlisten unter `/var/lib/apt/lists/` sind außerdem noch weitere generierte Metadaten zu den Paketen enthalten. Das beinhaltet bspw. die Debian Tags (siehe Kapitel 13), den Pfad und Dateinamen im Paketmirror, die Paketgröße und verschiedene Prüfsummen. Letztere dienen dazu, sicherzustellen, dass die Pakete fehlerfrei zwischen dem Paketmirror und ihrem Debian-System übertragen wurden und es zwischenzeitlich keine Veränderungen gab (siehe dazu Paketquelle überprüfen in Abschnitt 3.12 und Bezogenes Paket verifizieren in Abschnitt 8.35).

Das Paket *poppler-utils* umfasst beispielsweise die folgenden Metadaten:

```
Description-md5: cd43e3ed14322253876488d6f9911888
Tag: implemented-in::c++, interface::commandline, role::program,
 scope::utility, use::converting, use::filtering,
 works-with-format::pdf, works-with-format::xml, works-with::text
Filename: pool/main/p/poppler/poppler-utils_0.18.4-6_amd64.deb
Size: 162034
MD5sum: 0f0254920f85b6190ba7b03f4d2a7d73
SHA1: 77fb9d39145c60421462a8fe8315d0adaa49a38c
SHA256: 38f2d13ccddac9e3d05abff7c5fab353d3fea550c8f39293850651e03c3f8be4
```

4.2 Aufbau und Format

4.2.1 Generell: 2 Ebenen

Debianpakete beinhalten stets zwei Komponenten – Daten und Metainformationen. Die *Daten* sind die tatsächlichen Inhalte des Pakets, d.h. entweder der Quellcode oder die übersetzten Programmdateien, die bei der Installation auf Ihr System kopiert werden.

Die *Metainformationen* sind zusätzliche Informationen zu einem Paket, die dieses näher beschreiben. Dazu zählen erstens die Relationen zu anderen Paketen. Diese legen die Voraussetzungen fest, nach denen das Paket überhaupt übersetzt („gebaut“) und später auf Ihrem Linux-System installiert werden kann. Insbesondere zählen dazu die Abhängigkeiten und Konflikte zu anderen Paketen. Diese Relationen beschreiben wir ausführlich in Konzepte und Ideen dahinter (Abschnitt 4.1).

Zweitens gehören die sogenannten Maintainer-Skripte namens `preinst`, `postinst`, `prerm` und `postrm` dazu. Die beiden Skripte `preinst` und `postinst` regeln alle Aktivitäten vor und nach der Installation, `prerm` und `postrm` hingegen vor und nach der Entfernung des Pakets von Ihrem System. Diese vier Skripte sind üblicherweise distributionsspezifische Shellskripte, die der Maintainer des jeweiligen Pakets pflegt.

Für das Paket *dpkg-www* [Debian-Paket-dpkg-www] beinhaltet das bspw. das Starten des Webservers nach der Installation des Pakets, das Anhalten (vorher) und Neustarten (nachher) im Zuge einer Aktualisierung des Pakets und das Anhalten des Dienstes, bevor das Paket vom System wieder entfernt wird. In Abbildung 4.1 sehen Sie einen Ausschnitt aus dem `postinst`-Skript zu besagtem Paket.

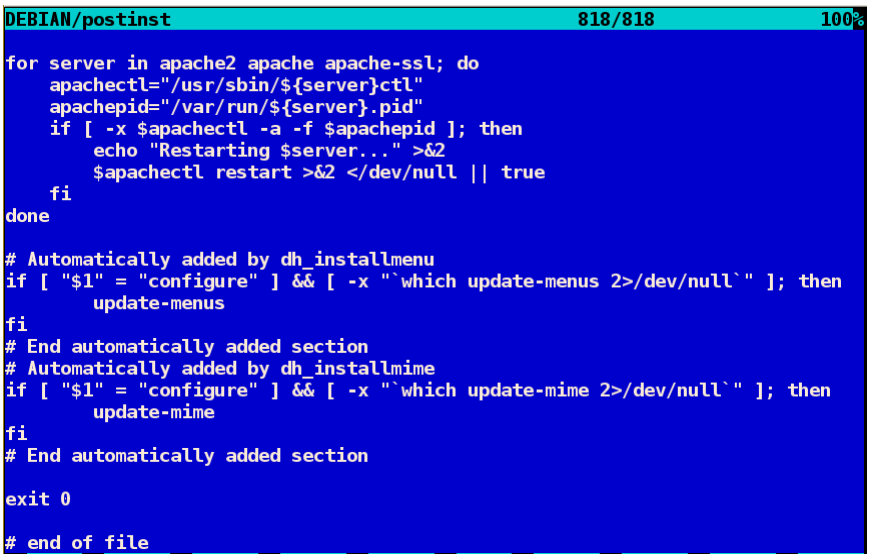

Abbildung 4.1: Auszug aus dem `postinst`-Skript zum Paket *dpkg-www*

4.2.2 Source-Pakete

Ein Source-Paket beinhaltet einerseits den Quellcode der Software und andererseits die Anweisungen, nach denen aus dem Quellcode der Software eines oder mehrere Binärpakete entstehen [Krafft-Debian-System]. Dazu besteht es aus mindestens 2, meistens jedoch 3 und ggf. auch noch weiteren Dateien:

`.dsc`
: Meta-Datei, die alle anderen Dateien mitsamt deren Hashsummen auflistet und ggf. signiert. Über die Hashsummen sind gleichzeitig alle anderen Dateien ebenfalls abgesichert.

Debian-spezifische Daten

- Bei Source-Format 1.0 ist es ein komprimierter Patch — erkennbar an der Erweiterung `diff.gz`. Dieser kann nur Textdateien enthalten.
- Bei Source-Format 3.0 ist es ein komprimiertes `tar`-Archiv — heutzutage meist mit `debian.tar.xz` benannt. Zulässig sind die Komprimierungsverfahren `gzip`, `bzip2`, `lzma` und `xz`. `lzma` wurde von Debian GNU/Linux allerdings nie unterstützt, wohl aber von anderen `deb`-basierten Distributionen (Abschnitt 1.5).
- Software, die nur im Debian-Paketformat vertrieben wird, verfügt weder über den komprimierten Patch noch das o.g. Archiv. In diesem Fall wird von sogenannten *nativen* Paketen gesprochen.

Upstream-Quellcode
: Der Originalquellcode der paketierten Software. Es wird versucht, diesen soweit wie möglich unverändert zu lassen. Es muss sich jedoch dabei um ein `tar`-Archiv handeln. Andere Container-Formate wie z.B. `zip` werden nicht unterstützt. Offerieren die Entwickler der Software den Quellcode z.B. nur als `zip`- oder `rar`-Archiv, so muss der Quellcode zunächst in ein `tar`-Archiv umgepackt werden. Der Dateiname des `tar`-Archivs endet dabei in `orig.tar.endung`, wobei das Suffix `endung` vom Komprimierungsformat abhängt. Erlaubt sind ebenfalls wieder `gzip`, `bzip2`, `lzma` und `xz`. `lzma` wird vom Debian-Archiv nicht unterstützt, aber von anderen `deb`-basierten Distributionen (Abschnitt 1.5). Ab dem Source-Format 3.0 kann ein Quellpaket (Abschnitt 2.7.4) mehr als ein `tar`-Archiv mit Upstream-Quellcode beinhalten. Diese haben dann die Endung `orig-komponente.tar.endung`, wobei `komponente` nur alphanumerische Zeichen und Bindestriche beinhalten darf.

Als **Paketformate** existieren die Versionen 1.0, 2.0 (wurde offiziell nie unterstützt) und 3.0 [Debian-DebSrc3.0]. Letzteres existiert in den zwei Varianten *quilt* (benannt als „3.0 (quilt)“) und *native* (benannt als „3.0 (native)“) und hat sich seit dessen Einführung mit Debian 6 *Squeeze* im Jahr 2011 mittlerweile etabliert. Dabei umfassen die Namen der Varianten für Version 3.0 jeweils auch die Leerzeichen und die beiden Klammern.

4.2.3 Binärpakete

Ein Debian-Binärpaket ist ein BSD-`ar`-Archiv, welches weitere, komprimierte `tar`-Archive beinhaltet. Nachfolgendes Beispiel zeigt das für das Paket *autotools-dev*.

Auspacken von Paketen mit ar

```
$ ar t autotools-dev_20100122.1_all.deb
debian-binary
control.tar.gz
data.tar.gz
$
```

Dabei stehen die einzelnen Komponenten eines Pakets für:

`debian-binary`
: Kennzeichnung für ein Debian-Paket. `debian-binary` ist eine Textdatei, welche lediglich die Versionsnummer des verwendeten Binär-Paketformats enthält. Nachfolgender Auszug zeigt die Versionsnummer für das Paket *mplayer*:

```
$ ar t mplayer_2%3a1.0~rc4.dfsg1+svn34540-1+b2_i386.deb
debian-binary
control.tar.gz
data.tar.gz
$ ar x mplayer_2%3a1.0~rc4.dfsg1+svn34540-1+b2_i386.deb debian-binary
$ cat debian-binary
2.0
$
```

`control.tar.gz`
: mit `gzip` komprimiertes `tar`-Archiv; dieses enthält die Kontrollinformationen für die Paketverwaltung

`data.tar.gz`, `data.tar.xz`, `data.tar.bz2`
: eigentliche Dateien des Pakets plus Speicherort, jeweils mit `gzip`, `xz` oder `bzip2` komprimiert

Ein Debian-Binärpaket ist eine Datei mit der Erweiterung `deb` oder `udeb`. Ersteres beinhaltet ausführbare Dateien, Daten, Dokumentation, Konfigurationsdateien und Copyright-Informationen [Krafft-Debian-System]. Bei `udeb`-Dateien handelt es sich um ein Paket mit reduziertem Paketinhalt, welches speziell für den Debian-Installer gedacht ist (siehe [Debian-udeb]).

Wie bereits oben angesprochen, beinhaltet jedes Debianpaket auch sogenannte *Control-Files* (nach [Krafft-Debian-System144]). Diese Steuerdateien werden in der Komponente `control.tar.gz` aufbewahrt und bestehen aus diesen Dateien:

`control`
: Das ist eine Steuerdatei und diese muss immer vorhanden sein. Sie beinhaltet die Metainformationen für die Paketverwaltung, bspw. zur Prüfung der Paketabhängigkeiten vor der Installation. Diese Steuerdatei kann beim Bauen des Pakets generiert worden sein, z.B. aus der Datei `control.in` mit Hilfe des Pakets *autotools*.

`conffiles`
: Das ist eine Liste mit Konfigurationsdateien zum Paket. Erfolgt eine Paketaktualisierung, werden die Dateien, die in dieser Liste aufgeführt sind, auf dem System beibehalten und nicht durch die Daten aus dem neuen Paket überschrieben. Damit bleiben bereits bestehende lokale Änderungen erhalten, bspw. von spezifisch angepassten Konfigurationsdateien. Diese Liste wird meist automatisiert generiert.

`preinst`
: Skriptdatei mit paketspezifischen Anweisungen. Diese Anweisungen werden *vor* der Installation oder Aktualisierung des Pakets (Upgrade) mit bestimmten Parametern aufgerufen.

`postinst`
: Skriptdatei mit paketspezifischen Anweisungen. Diese Anweisungen werden *nach* der Installation oder Aktualisierung (Upgrade) sowie zur Konfiguration des Pakets mit bestimmten Parametern aufgerufen.

prerm
Skriptdatei mit paketspezifischen Anweisungen. Diese Anweisungen werden mit bestimmten Parametern aufgerufen, *bevor* das Paket entfernt wird.

postrm
Skriptdatei mit paketspezifischen Anweisungen. Diese Anweisungen werden mit bestimmten Parametern aufgerufen, *nachdem* das Paket entfernt wurde.

md5sums
MD5-Summen der Dateien, welche im Paket enthalten sind. Damit wird sichergestellt, dass beispielsweise keine Übertragungsfehler (Bitfehler) oder Änderungen zwischen dem Paketmirror und ihrem lokalen System erfolgt sind (siehe auch Bezogenes Paket verifizieren in Abschnitt 8.35).

shlibs
Diese Datei listet Bibliotheken und *Shared Object Name* (kurz *SONAME*) auf, welches das Paket gemeinsam mit dem Paketnamen zur Verfügung stellt.

config
Skriptdatei. Diese erfragt vom Benutzer Konfigurationsparameter, welche für das Paket zur Einrichtung benötigt werden. Die Anworten werden direkt in der `debconf`-Datenbank abgelegt und bspw. im `postinst`-Skript verarbeitet.

templates
Diese Datei enthält Texte zu den Fragen und Hinweisen, die `debconf` während der Paketkonfiguration anzeigt (siehe dazu auch Pakete konfigurieren in Abschnitt 8.38).

Die eigentlichen Dateien zu einem Paket liegen in der *Datenkomponente*. Damit `dpkg` die zu installierenden Programme und Daten des Binärpakets auch an die richtige Position in der Dateisystemhierarchie kopieren kann, spiegelt der Inhalt dieser Komponente die entsprechende Verzeichnisstruktur auf dem Zielsystem vollständig wieder.

Diese Struktur, die zu installierenden Dateien sowie deren Typ und Größe, zeigen Sie mit dem Kommando `dpkg-deb -c Paketdatei` an. Am Beispiel von *vnstat* sehen Sie damit, dass im Debianpaket sowohl Programme (ausführbare Dateien in `/usr/bin` und `/usr/sbin`) als auch Dokumentation (in `/usr/share/doc` und `/usr/share/man`), Konfigurationsdateien (in `/etc`) und ein Verzeichnis für variable Daten (unterhalb von `/var/lib`) enthalten sind:

Inhalt des Pakets vnstat mit dpkg-deb anzeigen

```
$ dpkg-deb -c vnstat_1.10-1_i386.deb
drwxr-xr-x root/root         0 2010-04-20 20:38 ./
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/bin/
-rwxr-xr-x root/root    106424 2010-04-20 20:38 ./usr/bin/vnstat
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/sbin/
-rwxr-xr-x root/root     56184 2010-04-20 20:38 ./usr/sbin/vnstatd
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/share/
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/share/doc/
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/share/doc/vnstat/
-rw-r--r-- root/root      1604 2010-04-20 18:38 ./usr/share/doc/vnstat/changelog.Debian.gz
-rw-r--r-- root/root      2101 2010-01-02 01:32 ./usr/share/doc/vnstat/README
-rw-r--r-- root/root      3050 2010-01-02 02:36 ./usr/share/doc/vnstat/changelog.gz
-rw-r--r-- root/root      1501 2010-04-20 18:18 ./usr/share/doc/vnstat/copyright
-rw-r--r-- root/root      2077 2010-01-02 01:33 ./usr/share/doc/vnstat/FAQ.gz
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/share/man/
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/share/man/man1/
-rw-r--r-- root/root      2558 2010-04-20 20:38 ./usr/share/man/man1/vnstatd.1.gz
-rw-r--r-- root/root      4085 2010-04-20 20:38 ./usr/share/man/man1/vnstat.1.gz
drwxr-xr-x root/root         0 2010-04-20 20:38 ./usr/share/man/man5/
-rw-r--r-- root/root      2488 2010-04-20 20:38 ./usr/share/man/man5/vnstat.conf.5.gz
drwxr-xr-x root/root         0 2010-04-20 20:38 ./etc/
drwxr-xr-x root/root         0 2010-04-20 20:38 ./etc/init.d/
-rwxr-xr-x root/root      1466 2010-04-20 17:52 ./etc/init.d/vnstat
-rw-r--r-- root/root      2889 2010-04-20 20:38 ./etc/vnstat.conf
drwxr-xr-x root/root         0 2010-04-20 20:38 ./var/
```

```
drwxr-xr-x root/root         0 2010-04-20 20:38 ./var/lib/
drwxr-xr-x root/root         0 2010-04-20 20:38 ./var/lib/vnstat/
$
```

4.2.4 Übergangs- und Metapakete

Wie bereits in Übergangs- und Metapakete (siehe Abschnitt 2.7.2) deutlich wurde, handelt es sich hierbei um Binärpakete, die eine spezielle Charakteristik haben: sie haben meist außer der Dokumentation keine weiteren Inhalte. Der eigentliche Inhalt sowie Sinn und Zweck sind die Beschreibung der Abhängigkeiten der Pakete.

Übergangspakete werden auch *Dummypakete* oder *Transitionspakete* genannt. Deren Aufgabe ist es, Paketumbenennungen bei der Aktualisierung auf eine neue Veröffentlichung sauber zu handhaben und in diesem Zusammenhang auftretende Abhängigkeitskonflikte zu verhindern. Metapakete erleichtern dagegen nur die Installation einer Gruppe von zusammenhängenden Paketen.

Ein Paket dieser Art beinhaltet meist nur zwei Dateien unterhalb von `/usr/share/doc` — die Informationen zum Copyright und die bisherigen Änderungen. Letzteres liegt in der Datei `changelog.Debian.gz`. Beide Dateien können aus Gründen der Platzersparnis durch einen symbolischen Link auf eine der Abhängigkeiten ersetzt werden, falls diese aus dem gleichen Sourcepaket gebaut wurden.

Darüberhinaus können die Pakete einen Wrapper oder einen symbolischen Link zur Wahrung der Rückwärtskompatibilität beinhalten. Beispielsweise umfasst das Paket *ash* nur eine Abhängigkeit auf das Paket *dash* und einen symbolischen Verweis (Symlink) von `/bin/ash` zu `/bin/dash`:

Symbolischer Verweis auf eine andere Komponente am Beispiel der ash

```
$ ls -la /bin/ash
lrwxrwxrwx 1 root root 4 Mär  1  2012 /bin/ash -> dash
$
```

Kapitel 5

APT und Bibliotheken

Wie bereits in der Übersicht in Softwarestapel und Ebenen (siehe Abschnitt 2.3) deutlich wurde, ist die Paketverwaltung von Debian GNU/Linux mehrstufig und modular aufgebaut. Hinter den Bedienoberflächen `dpkg`, APT und `aptitude` (siehe Werkzeuge zur Paketverwaltung in Kapitel 6) stecken mächtige Bibliotheken, die den Zugriff auf die einzelnen Softwarepakete und die Paketdatenbank kapseln. Mit Hilfe der nachfolgend vorgestellten Bibliotheken und den Funktionen daraus können Sie eigene Anwendungen zur Paketverwaltung entwickeln.

5.1 Bibliothek `libapt-pkg`

Diese Bibliothek aus dem Paket *libapt-pkg4.12* [Debian-Paket-libapt-pkg4.12] enthält die Basiskomponenten zum Zugriff auf die einzelnen Softwarepakete. Das umfasst Funktionen zur Suche nach Paketen, deren Verwaltung sowie die Ausgabe der Paketinformationen. Dazu gehören:

- der Abruf von Informationen zu einem Paket aus den verschiedenen Paketquellen
- der Abruf eines Pakets und der vollständigen Auflösung der Paketabhängigkeiten dieses Pakets
- die Authentifizierung der Paketquellen und Überprüfung der abgerufenen Daten (Validierung)
- die Installation und Entfernung von Paketen aus ihrem Linux-System
- der Zugriff auf den Paketcache (siehe Kapitel 7)
- die Bereitstellung von Schnittstellen zu Netzwerkprotokollen, um Daten und Pakete über diese beziehen zu können. Dazu gehören bspw. CD-ROM, FTP, HTTP/S, rsh und DebTorrent (APT über BitTorent) ([Debian-Paket-apt-transport-debtorrent] und [Debian-Wiki-DebTorrent]).

5.2 Bibliothek `libapt-pkg-perl`

Diese Bibliothek aus dem Paket *libapt-pkg-perl* [Debian-Paket-libapt-pkg-perl] beinhaltet die Perl-Schnittstelle zum Zugriff auf die einzelnen Softwarepakete. Es hat die gleiche Funktionalität wie das weiter oben beschriebene Paket *libapt-pkg*.

5.3 Bibliothek `libapt-pkg-doc`

Dieses Paket [Debian-Paket-libapt-pkg-doc] stellt die Dokumentation zur Verfügung, auf deren Grundlage Sie eigene Entwicklungen in APT anstoßen können. Die Dokumentation steht als Plaintext und als HTML-Dokument bereit.

5.4 Bibliothek `libapt-inst`

Um Informationen aus `deb`-Paketen zur erhalten, nutzen Sie diese Bibliothek aus dem Paket *libapt-inst* [Debian-Paket-libapt-inst]. Darüber steht eine Schnittstelle zur Abfrage der Paketinterna bereit, die sowohl den Paketinhalt, als auch die Steuerdaten der Komponente `control.tar.gz` umfassen (siehe Debian-Paketformat im Detail in Abschnitt 4.2).

Kapitel 6

Werkzeuge zur Paketverwaltung (Überblick)

6.1 Frontends für das Paketmanagement

Unter einem *Frontend* verstehen wir ein Programm oder ein Werkzeug mit einer Bedienoberfläche, welches im Alltag von Ihnen für die Verwaltung der Softwarepakete verwendet wird. Es deckt alle dafür notwendigen Aktionen auf ihrem System ab und umfasst die grundsätzliche Pflege des Paketbestands. Dazu zählen bspw. die Installation, die Aktualisierung und die restlose Entfernung von Softwarepaketen, wobei das Gesamtsystem stets in einem konsistenten, benutzbaren Zustand verbleibt.

Frontends existieren in recht unterschiedlichen Varianten und folgen divergierenden Bedienkonzepten. Die nachfolgende Übersicht orientiert sich daher an der Benutzerschnittstelle und dem Paketformat, für das Sie das jeweilige Programm benutzen können. Einige Programme sind Zwitter und stellen mehrere Bedienmodi zur Verfügung, so bspw. SmartPM (Abschnitt 6.4.3), welches Sie sowohl über die Kommandozeile, als auch über eine graphische Oberfläche (GUI) bedienen können. `aptitude` (Abschnitt 6.3.2), `aptsh` (Abschnitt 6.2.3), `cupt` (Abschnitt 6.2.5) und `wajig` (Abschnitt 8.42.7) stellen über die Kommandozeile hinaus auch ein eigenes Text User Interface (TUI) bereit. Die nachfolgende Zusammenstellung in Tabelle 6.1 ist daher nicht ganz diskussionsfrei und erhebt zudem keinen Anspruch auf Vollständigkeit.

Tabelle 6.1: Frontends zur Paketverwaltung

Kategorie	**deb-basierte Systeme**	**rpm-basierte Systeme**	**andere Paketformate**
Kommandozeile	`dpkg`, `dpkg-www`, APT, `aptitude`, `aptsh`, `cupt`, SmartPM, `gdebi` (*gdebi-core*), `wajig`	`rpm`, `yum`, `urpmi`, `zypper`, SmartPM	`emerge`, `pacman`
Text User Interface (TUI)	`tasksel`, `aptitude`, Debian Installer, Univention Installer für Univention Corporate Server (UCS)	Yet another Setup Tool (YaST), DrakConf oder Mandriva Linux Control Center (MCC) [Mandriva-Wiki] bzw. Mageia Control Center (MCC) (Textkonsole)	pcurses
Graphical User Interface (GUI)	Synaptic, SmartPM, Ubuntu Software Center, PackageKit, Apper (früher *KPackageKit*), `gdebi`	Yet another Setup Tool 2 (YaST2), DrakConf oder Mandriva Linux Control Center (MCC) [Mandriva-Wiki] bzw. Mageia Control Center (MCC)	PacmanXG4, PacmanExpress, tkPacman, GNOME PackageKit, Zenity Pacman GUI, Octopi

Tabelle 6.1: (continued)

Kategorie	deb-basierte Systeme	rpm-basierte Systeme	andere Paketformate
webbasierte Verwaltung (WUI)	IP Brick [ipbrick], Univention Management Console für Univention Corporate Server (UCS), Ubuntu Landscape , Appnr, Communtu, Debian Pure Blends		

6.1.1 Aufgaben, Sinn und Zweck des Frontends

Basierend auf der Einordnung in die unterschiedlichen Softwarestapel und Ebenen (siehe Abschnitt 2.3) lässt sich der Aufgabenbereich und damit der Funktionsumfang eines Programms zur Paketverwaltung konkreter fassen. Dabei kommen häufig die UNIX-Prinzipien „Ein Werkzeug für eine Aufgabe“ und „Keep it simple, stupid“ (sinngemäß: Mach's so einfach wie möglich) sehr stark zum tragen.

Zur *unteren Ebene* gehört das Programm `dpkg`. Es bietet grundsätzliche Funktionen, die ein erforderliches Minimum abdecken. Die Funktionen betreffen nur das lokale System und setzen voraus, dass alle notwendigen Informationen und `deb`-Pakete bereits vorliegen. Dazu gehören die Fähigkeiten, Informationen über installierte und noch zur Verfügung stehende Pakete und Paketdateien anzuzeigen sowie bereits lokal als Datei vorliegende Pakete zu installieren, zu konfigurieren und wieder vom System zu entfernen. `dpkg` fokussiert dabei eher auf Einzelpakete, bspw. der Aufruf `dpkg -i Paketname` zur Installation eines Pakets (siehe auch Abschnitt 8.36).

Die *obere Ebene* beinhaltet im weitesten Sinne alle übergeordneten Aufgaben, wie bspw. komplexere Verwaltungsfunktionen. Dazu zählt das Herunterladen der Paketlisten von den vorher von Ihnen festgelegten Paketmirrors, das Aktualisieren der lokalen Paketlisten, das Auswählen und Beziehen eines Pakets von einem passenden Paketmirror, das Auflösen der Paketabhängigkeiten und das Klären weiterer, dazu benötigter oder empfohlener Pakete, die zum ausgewählten Paket passen und welche für Sie als Benutzer interessant sein könnten. Ebenso gehört die Validierung eines Pakets anhand seines GPG-Schlüssels (siehe Abschnitt 8.35) dazu. Zur oberen Ebene zählen bspw. Programme wie `tasksel`, APT, `aptitude`, SmartPM, das Ubuntu Software Center und die PackageKit-Varianten *apper* (KDE) und *gnome-packagekit* (GNOME).

Eine Mischform stellen hingegen die Programme `aptsh`, `cupt`, `wajig` und `gdebi` dar. Deren Anspruch ist es, beide Ebenen in einem einzigen Programm abzudecken und alle erforderlichen Funktionen zur Paketverwaltung bereitzustellen. Die genannten Programme kommen diesem Ziel derzeit in unterschiedlicher Qualität nahe. Dabei erfolgt ein Zugriff auf die bestehenden Bibliotheken, der durch eigene, zusätzliche Funktionalitäten ergänzt wird.

6.1.2 Anmerkungen zur Programmauswahl

Es gibt keine Regelung oder Empfehlung dafür, welches Programm aus obiger Liste Sie benutzen sollen. Dafür sind die Wissensstände, Gewohnheiten und Vorlieben im Umgang mit Software zu unterschiedlich (siehe auch „Ausblick und Empfehlungen für Einsteiger“ in Abschnitt 44.1).

In der Praxis zeigt sich, dass `apt-get` häufig die schnellste und effizienteste Variante ist, sofern Sie den exakten Namen eines Debian-Pakets (siehe dazu Abschnitt 2.11) oder zumindest einen Großteil davon wissen. Die Kommandozeilenwerkzeuge sind sehr flexibel und verfügen über eine hohe Anzahl von Funktionen. Diese sprechen Sie über vielfältige Unterkommandos, Schalter und Parameter an.

Viele Schalter und Parameter der Kommandozeilenwerkzeuge werden in den TUI, GUI und WUI nicht oder nur unzureichend abgebildet, sind zudem in den meisten Fällen geschickt versteckt, anders benannt oder auch mitunter sinnentstellend übersetzt. Das sorgt vielfach für Unmut und Verzweiflung bei der Suche nach einer bestimmten Funktionalität. Erfahrenere Benutzer vermissen häufig die Flexibilität der vielen Optionen und greifen daher bevorzugt zur Kommandozeile oder zum TUI, da das schneller und einfacher geht. Das hoffnungs- und erwartungsvolle Herumklicken in einer graphischen Anwendung möchten sie den Marketingfritzen und Mausschubsern überlassen.

Die Komplexität der Kommandozeilenwerkzeuge kann Einsteiger überfordern — gleiches gilt aber auch für graphische Oberflächen. In jedem Fall setzt es bei Ihnen den Willen zur Einarbeitung voraus — gleich welches Werkzeug es auch ist. Der Vorteil der Kommandozeilenwerkzeuge liegt darin, dass sie meist zur Basisinstallation Ihres Debian-Systems gehören und somit auch auf ferngewarteten Serversystemen zur Verfügung stehen. Graphische Werkzeuge sind in der Regel nur auf Desktopsystemen installiert. Webbasierte Benutzerschnittstellen sind deutlich in der Minderheit und haben den Exotenstatus. Steigt der Verbreitungsgrad UNIX/Linux-basierter Smartphones und TabletPCs mit Android bzw. Ubuntu weiter an, ist mit einer Zunahme von Programmen wie Appnr (siehe Abschnitt 6.5.3) im Alltag zu rechnen.

6.2 Für die Kommandozeile

6.2.1 dpkg

`dpkg` ist das Debian-Programm für grundlegende Paketoperationen und bildet in Bezug auf Funktionsumfang und Handhabung das Äquivalent zu `rpm` auf RedHat-basierten Linuxsystemen. Es kürzt den Namen *Debian GNU/Linux package manager* ab. Im Anhang unter ``Kommandos zur Paketverwaltung im Vergleich`` (siehe Kapitel 46) stellen wir die verschiedenen Schalter zu den beiden Kommandos `dpkg` und `rpm` gegenüber.

`dpkg` agiert nur mit Paketen, die schon auf ihrem Linuxsystem lokal vorliegen — entweder als `deb`-Datei in einem Verzeichnis oder als bereits installiertes Paket. `dpkg` kann keine Pakete von einem Paketmirror beziehen.

Sie erreichen `dpkg` ausschließlich über die Kommandozeile und starten es mit diversen Schaltern und Optionen. Die wichtigsten Parameter für den Gebrauch im Alltag sind[1]:

* Paketliste ausgeben (Abschnitt 8.5) (`dpkg -l`)

- Paketstatus erfragen (Abschnitt 8.4) mit `dpkg -s` *Paketname*
- Inhalt eines installierten Pakets anzeigen (Abschnitt 8.23) mit `dpkg -L` *Paketname*
- Inhalt eines nicht installierten Pakets anzeigen (Abschnitt 8.23) mit `dpkg -c` *Paketname*
- Paket zu Datei finden (Abschnitt 8.22) mit `dpkg -S` *Paketname* und
- Pakete konfigurieren (Abschnitt 8.38) (Option `--configure`)

Mit `dpkg` zeigen Sie die installierten Pakete und deren Zustand an, suchen nach Paketinhalten und konfigurieren im Bedarfsfall ein Paket nach.

Für alle anderen Aktionen sind hingegen die Werkzeuge `apt-get` (Abschnitt 6.2.2), `apt-cache`, `aptitude` (Abschnitt 6.3.2) und `apt-file` oder die Benutzeroberflächen via Ncurses oder GTK besser geeignet (siehe Abschnitt 6.3 und Abschnitt 6.4). Diese fassen viele Einzelschritte von `dpkg` zusammen und vereinfachen Ihnen die Wartung ihres Systems erheblich.

6.2.2 APT

APT ist das Debian-Programm für etwas komplexere Paketoperationen und steht als Abkürzung für *Advanced Packaging Tool*. Sie finden es im Paket *apt* [Debian-Paket-apt], welches zur Standardinstallation Ihres Debian-Systems gehört.

APT ist für den Alltagseinsatz konzipiert. Es eignet sich sowohl für Recherchezwecke (Abfrage von Status- und Zustandsinformationen), als auch für die Installation und Aktualisierung einzelner Pakete sowie gesamter Paketstrukturen (Veröffentlichungen).

Im Gegensatz zu `aptitude` (siehe Abschnitt 6.3.2) ist es deutlich weniger anspruchsvoll. Das betrifft die Anforderungen an die Hardware und insbesondere den benötigten Speicher für die Ausführung. APT hat zudem eine deutlich höhere Ausführungsgeschwindigkeit als `aptitude`.

APT ist sehr mächtig und kann mit Paketen umgehen, die sich entweder bereits lokal auf Ihrem System befinden, oder noch auf einem Paketmirror vorliegen. Es kombiniert i.d.R. mehrere Einzelaktionen von `dpkg`, greift dabei aber nicht direkt auf `dpkg` zurück, sondern kapselt dafür die Aufrufe mit Hilfe der Bibliothek `libapt-pkg` (siehe dazu APT und Bibliotheken unter Kapitel 5).

[1] Weitere Optionen zu `dpkg` entnehmen Sie bitte der Manpage zum Programm

APT umfasst ausschließlich Programme für die Kommandozeile. Dazu zählen `apt-cache`, `apt-cdrom` (siehe Abschnitt 3.8), `apt-config` zur Konfiguration von APT (siehe Kapitel 10), `apt-get`, `apt-key` (siehe Abschnitt 3.12) und `apt-mark` (siehe Abschnitt 8.4.6). Jedes der genannten Programme verfügt über umfangreiche Unterkommandos, die Sie wiederum mit diversen Optionen und Schaltern kombinieren können. Die gebräuchlichsten Aktionen für den Alltag sind:

- Paketstatus erfragen (Abschnitt 8.4) mit `apt-cache show` *Paketname*
- Inhalt eines Pakets anzeigen (Abschnitt 8.23) mit `apt-file show` *Paketname*
- Paketabhängigkeiten anzeigen (Abschnitt 8.17) mit `apt-cache depends` *Paketname*
- Paket über den Namen oder die Beschreibung finden (Abschnitt 8.19) mit `apt-cache search` *Paketname*
- Paket installieren (Abschnitt 8.36) mit `apt-get install` *Paketname*
- Installierte Pakete löschen (Abschnitt 8.41 mit `apt-get remove` *Paketname*
- Paketliste aktualisieren (Abschnitt 3.13) mit `apt-get` *update*
- neuere Versionen für die Pakete einspielen (Abschnitt 8.39) mit `apt-get upgrade`)
- die gesamte Distribution aktualisieren (Abschnitt 8.45) mit `apt-get dist-upgrade`

Nachfolgend geben wir Ihnen eine Übersicht zu allen Unterkommandos, die die einzelnen APT-Werkzeuge bereithalten. Neben dem jeweiligen Unterkommando finden Sie den Verweis auf den entsprechenden Abschnitt im Buch, in dem wir auf dieses genauer eingehen.

`apt-cache` bietet die folgenden Unterkommandos:

`depends`
: Paketabhängigkeiten anzeigen (siehe Abschnitt 8.17)

`dotty`
: einen Abhängigkeitsgraphen im `dot`-Format für die benannten Pakete erzeugen (siehe das Beispiel in Abschnitt 2.5)

`dump`
: eine kurze Programminformation von jedem Paket im Paketcache anzeigen

`dumpavail`
: die Liste der verfügbaren Pakete anzeigen

`gencaches`
: den Paketzwischenspeicher von APT erzeugen

`madison`
: verfügbare Versionen eines Pakets anzeigen (siehe Abschnitt 8.18 und Abschnitt 8.18.2)

`pkgnames`
: die Namen aller Pakete auflisten, die APT kennt (siehe Abschnitt 8.3)

`policy`
: die Quellen und deren Prioritäten auflisten (siehe Abschnitt 8.18)

`rdepends`
: umgekehrte Paketabhängigkeiten anzeigen (siehe Abschnitt 8.17)

`search`
: Paket über den Namen finden (siehe Abschnitt 8.19)

`show`
: Paketinformationen ausgeben und Paketstatus erfragen siehe (Abschnitt 8.4)

`showsrc`
: Informationen zum Sourcepaket anzeigen (siehe Abschnitt 8.34)

showpkg
Informationen über das Paket anzeigen (siehe Abschnitt 8.4)

stats
Statistik zum Paketcache ausgeben (siehe Abschnitt 7.2)

unmet
eine Zusammenfassung aller unerfüllten Abhängigkeiten im Paketcache ausgeben (siehe Paketstatus erfragen in Abschnitt 8.4)

xvcg
einen Abhängigkeitsgraphen für *xvcg* für die benannten Pakete erzeugen

`apt-get` gehört mit Sicherheit zur Menge der gebräuchlichsten Kommandos der APT-Familie und verfügt über die folgenden Unterkommandos:

autoclean
Paketcache aufräumen (siehe Abschnitt 7.3)

autoremove
Paketwaisen löschen (siehe Abschnitt 8.42)

build-dep
Abhängigkeiten eines Sourcepakets erfüllen (findet Verwendung beim Erstellen von Paketen)

check
Paketcache auf beschädigte Paketabhängigkeiten prüfen (siehe Abschnitt 8.17)

clean
Paketcache aufräumen (siehe Abschnitt 7.3)

dist-upgrade
Distribution aktualisieren (siehe Abschnitt 8.45)

download
Paketdatei nur herunterladen (siehe Abschnitt 8.31)

dselect-upgrade
Aktualisierung der Pakete über `dselect`

install
Paket installieren (siehe Abschnitt 8.36)

purge
Paket inklusive Konfigurationsdateien des Pakets entfernen (siehe Abschnitt 8.41)

remove
Paket deinstallieren (siehe Abschnitt 8.41)

source
Beziehen der Sourcepakete (siehe Abschnitt 8.33)

update
Paketliste aktualisieren (siehe Abschnitt 3.13)

upgrade
Pakete auf eine neue Version aktualisieren (siehe Abschnitt 8.39)

Für `apt-key` sind die Unterkommandos `add`, `adv`, `del`, `export`, `exportall`, `finger`, `list`, `net-update` und `update` zulässig. Diese besprechen wir ausführlich unter „Paketquelle auf Echtheit überprüfen“ in Abschnitt 3.12.

Die Unterkommandos von `apt-mark` lauten `auto`, `manual`, `showauto` und `showmanual`. Dazu gehen wir unter „Paketstatus erfragen“ in Abschnitt 8.4 detailliert ein.

Die **Weiterentwicklung von APT** geht stetig voran. Seit mehreren Jahren gibt es Bestrebungen, APT grundlegend zu erneuern bzw. dessen verteilte Funktionalität unter einer einzigen Benutzeroberfläche zusammenzufassen. Unter dem Namen APT2 [apt2] existiert zwar ein Prototyp mit neuer API, jedoch gab es dort nach unserer Recherche seit 2011 keine weitere Entwicklung mehr.

Eine weniger tiefgreifende, aber dennoch erfrischende Modernisierung gibt es seit **APT Version 1.0**. Von da an enthält das Paket *apt* das zusätzliche, gleichnamige Kommandozeilenprogramm `apt`. Dieser Programmname wurde bis dato von einem Java-Programm zur Annotationsverarbeitung (*Annotation Processing Tool*) belegt [Java-Apt]. Es wird seit Java 7 als *veraltet* deklariert und ist seit Java 8 nicht mehr Bestandteil von Java.

Somit wurde der Weg für ein neues Programm frei, ohne große Verwirrung zu stiften. `apt` vereint die gängigsten Unterkommandos von `apt-get` und `apt-cache` in einem kürzeren Befehl und mit moderneren Standardeinstellungen wie z.B. einem Fortschrittsbalken und farbiger Ausgabe auf dem Terminal (siehe [Vogt-Apt-1.0]). Neben den bekannten Unterkommandos `list`, `search`, `show`, `update`, `install` und `upgrade` kennt es auch die neuen Aktionen `full-upgrade` als Ersatz für `dist-upgrade` und `edit-sources` zur direkten Veränderung der Datei `/etc/apt/sources.list` (siehe Abschnitt 8.39 und Abschnitt 3.3). Darüber hinaus verfügt es ab **APT Version 1.1** über die Fähigkeit, lokal vorliegende `deb`-Pakete zu installieren und dabei die dazugehörigen Paketabhängigkeiten mit zu berücksichtigen.[2]

Ebenfalls in produktivem Zustand und teilweise intensiver Benutzung befinden sich die Werkzeuge `aptsh`, `cupt`, `aptitude` und SmartPM. Während sich `aptsh` und `cupt` nur auf die Kommandozeile beschränken, bieten Ihnen `aptitude` zusätzlich eine textbasierte bzw. SmartPM eine graphische Benutzeroberfläche. Auf diese Werkzeuge gehen wir nachfolgend genauer ein (siehe Abschnitt 6.2.3, Abschnitt 6.2.5, Abschnitt 6.3.2 und Abschnitt 6.4.3).

6.2.3 Die `aptsh`

Bei der `aptsh` handelt es sich um eine Shell für die Paketverwaltung. Die Grundlage dafür bildet die Bibliothek *libapt-dpkg* (siehe Kapitel 5). Als Inspirationsquelle für die Shellkommandos dienen die Unterkommandos von APT und `aptitude`, die hiermit in einem Werkzeug zusammengefasst wurden, somit die Handhabung erleichtern und den Tippaufwand deutlich verringern.

Vom Umfang her kombiniert die `aptsh` eine Vielzahl der Unterkommandos von `apt-get`, `apt-cache` und `aptitude`. Einerseits agiert es als Shell, andererseits kann es die Aufrufe mit den Optionen über die Kommandozeile direkt verarbeiten. Zu den üblichen Kommandos zählen bspw. `install`, `remove`, `purge`, `search`, `update` und `upgrade`. Weiterhin sind auch zur Paketsuche `ls` und `rls` verfügbar, ebenso `depends` und `rdepends` für die Anzeige der Paketabhängigkeiten (siehe Abbildung 6.1) sowie `changelog`, `show`, `showpkg`, `showsrc` und `whatis` für Informationen zu einem Paket.

Abbildung 6.1: `aptsh` mit der Ausgabe der umgekehrten Paketabhängigkeiten mit `rdepends`

2012 probierten Thomas Winde und Frank Hofmann die `aptsh` aus und waren von dem Werkzeug in kürzester Zeit hellauf begeistert. Dazu entstand zunächst ein Beitrag im Magazin *Linux User* [Hofmann-Winde-Aptsh-LinuxUser], der die kleine Entdeckungsreise mit dem Programm beschreibt. Erst später erschienen andere Möglichkeiten wie `wajig` [Debian-Paket-wajig] und `cupt` [Debian-Paket-cupt] auf unserem Radar, welche wir dann für das vorliegende Buch genauer inspizierten (siehe nachfolgende Abschnitte).

Hintergrund für die Recherche nach Alternativen zur `aptsh` war der Fakt, dass das Projekt seit Debian 6 *Squeeze* leider verwaist ist. Alle Pakete sind lediglich Non-Maintainer-Uploads (NMU) und damit ohne kontinuierliche Betreuung. Zudem kommt hinzu, dass das Projekt [aptsh-Projekt] seit dem Auf-und-Ab von Berlios unter wechselnden Adressen auffindbar ist. Das vereinfacht die Pflege nicht unbedingt.

[2] Diese Eigenschaft stammt vom Programm `gdebi` (siehe Abschnitt 6.4.6), welches ebenfalls vom APT-Entwickler Michael Vogt gepflegt wird.

6.2.4 wajig

Das in der Programmiersprache Python geschriebene Programm *wajig* [Debian-Paket-wajig] ist vorrangig ein Wrapper um `dpkg` (Abschnitt 6.2.1) und APT (Abschnitt 6.2.2). Es zählt zur gleichen Kategorie wie die `aptsh` (siehe Abschnitt 6.2.3), beinhaltet aber auch Elemente von `cupt` (Abschnitt 6.2.5) und `aptitude` (Abschnitt 6.3.2) auf der Kommandozeile. Bis einschließlich Debian 6 *Squeeze* bestand zudem eine graphische Variante namens *gjig*, die mittlerweile obsolet ist.

`wajig` zielt darauf ab, alle im Alltag erforderlichen Aktionen zur Paketverwaltung in *einem* Werkzeug für die Kommandozeile zusammenzufassen. Daher haben sich die *wajig*-Entwickler das Ziel gesetzt, die APT-Bibliotheken (siehe Kapitel 5) vollständig auszureizen und nach Möglichkeit auch alle Optionen, die `dpkg` und APT bieten, im Programm zu integrieren. Gleichzeitig stehen auch Funktionen bereit, die von den separaten Werkzeugen wie bspw. `apt-cdrom` (Abschnitt 3.8) oder `alien` (siehe Abschnitt 22.2) entlehnt wurden.

Sie bedienen `wajig` ausschließlich über die Tastatur. Möglich sind zwei Modi — mit dem gewünschten Unterkommando beim Aufruf, oder ohne. Bei ersterem erfolgt die Ausgabe direkt im Terminal, bei letzterem öffnet sich dann zunächst die `wajig`-Shell und wartet auf Ihre Eingabe. In dieser können Sie dann alle Unterkommandos zur Paketverwaltung benutzen. Dazu zählen bspw. `install` zur Paketinstallation, `detail` zur Darstellung der Paketinformationen, `listfiles` zu Auflistung des Paketinhalts und `remove` zum Entfernen eines Pakets. Mittels `find-file` erstöbern Sie eine gewünschte Datei in den bereits installierten Paketen, wohingegen Ihnen `list-orphans` die Paketwaisen (siehe Abschnitt 8.42) anzeigt.

Als Besonderheit ist einerseits die Anbindung an *apt-get.org* [apt-get.org] zu nennen, um den Paketmirror dynamisch auszuwählen (siehe Abschnitt 3.4.3). Ebenso ist die Umwandlung und Installation von `.rpm`-Paketen mittels `rpm2deb` und `rpminstall` sowie die ausführliche, integrierte Hilfe hervorzuheben.

Suche nach der Datei sources.list mit Hilfe von wajig

```
$ wajig find-file sources.list
apt: /usr/share/man/es/man5/sources.list.5.gz
apt: /usr/share/man/ja/man5/sources.list.5.gz
apt: /usr/share/man/pt/man5/sources.list.5.gz
debtags: /etc/debtags/sources.list
apt: /usr/share/man/fr/man5/sources.list.5.gz
apt: /usr/share/doc/apt/examples/sources.list
debtags: /etc/debtags/sources.list.d
apt: /usr/share/man/de/man5/sources.list.5.gz
debtags: /etc/debtags/sources.list.d/source-example
apt: /usr/share/man/pl/man5/sources.list.5.gz
apt: /etc/apt/sources.list.d
apt: /usr/share/man/man5/sources.list.5.gz
$
```

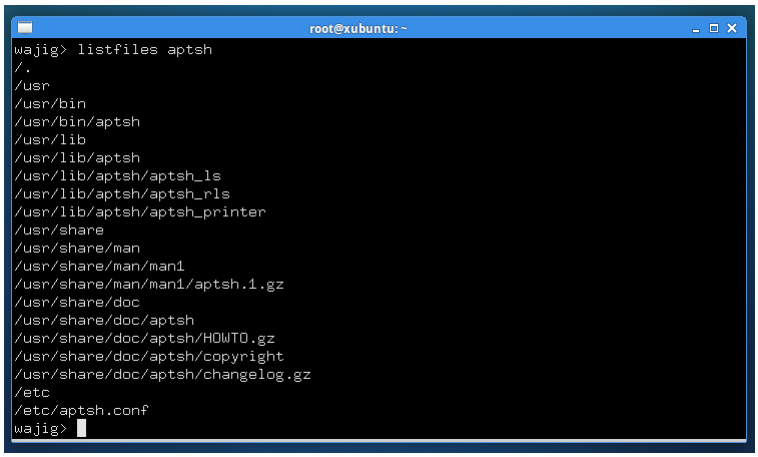

Abbildung 6.2: `wajig` mit der Ausgabe des Kommandos `listfiles`

Weitere Informationen zum Programm finden Sie auf der Webseite des Projekts [wajig-Webseite]. Um die Feinheiten der Kommandos zwischen `dpkg`, APT und `wajig` besser vergleichen zu können, hilft ein Blick in das Wiki von xtronics [xtronics-Wiki].

6.2.5 Cupt

Cupt beschreibt sich selbst als *High-level Package Manager* und integriert Kommandos unter einem Dach, die Sie von den Werkzeugen `dpkg` und APT her kennen. Dafür nutzt es auf der Serverseite die gleiche Infrastruktur wie APT. Die Clientseite wurde hingegen komplett neu entwickelt. Sie rufen das in der Programmiersprache C++ entwickelte Werkzeug über das gleichnamige Kommando `cupt` auf.

Wie bereits oben angerissen, vereinigt Cupt zwar Kommandos aus `dpkg` und APT, jedoch bislang noch nicht alle davon. Offen ist bspw. der Support für *multiarch* (Abschnitt 1.2.3). Gleichzeitig bietet es auch weitere Features, die APT noch fehlen (siehe [Debian-Wiki-cupt]), bspw. Debdelta [Debdelta] und die Synchronisation anhand der Version des Sourcepakets. Ebenso kennt es ein Kommando `satisfy`, um auf der Kommandozeile angegebene Paketbedingungen zu erfüllen, z.B. `cupt satisfy ``kmail (>=4:4.2), wget (>=1.10.0)"`.

Cupt kann problemlos parallel zu APT verwendet werden, ist jedoch gemäß seinem Autor noch nicht sehr weit verbreitet und auch entsprechend wenig durch Benutzer in der Praxis getestet [Cupt-Tutorial]. Wir gehen im Buch nicht weiter darauf ein.

6.3 ncurses-basierte Programme

6.3.1 tasksel

`tasksel` gehört zu den Anwendungen, die Sie vielleicht nur aus der textbasierten Installation von Debian her kennen. Nach der Zusammenstellung des Debian-Basissystems wird dieses Werkzeug üblicherweise einmal automatisch im Installationsprozess aufgerufen und gerät danach vollständig in Vergessenheit. Stattdessen helfen Ihnen APT und `aptitude` bei den Routineaufgaben.

Der Name ist eine Abkürzung und steht für *task select*, auf Deutsch übersetzbar mit „Aufgabe auswählen“. Das Paket *tasksel* [Debian-Paket-tasksel] beinhaltet lediglich die Benutzeroberfläche, das Paket *tasksel-data* hingegen eine Liste mit vorab festgelegten Standardaufgaben (englisch *tasks*). Jeder genannten Aufgabe sind eine Reihe von Paketen zugeordnet.

Die beiden *tasksel*-Generationen 2.x und 3.x unterscheiden sich massiv voneinander. Während Generation 2 noch von `aptitude` abhängt, setzt Generation 3 hingegen verstärkt auf die Nutzung von Metapaketen (siehe Abschnitt 2.7.2). Das zeigt sich sehr deutlich in den Ausgaben im Terminal, auf die wir unten genauer eingehen.

Abbildung 6.3: Softwareauswahl in `tasksel`

Über die textbasierte Benutzeroberfläche und der dargestellten Liste wählen Sie zunächst mittels Pfeil- und Leertaste die gewünschten Aufgaben aus. Daraufhin werden alle Pakete „in einem Rutsch“ auf Ihrem Linuxsystem installiert, die diesen Aufgaben zugeordnet sind.

Bei Debian und Ubuntu existieren viele Aufgaben als separate, vorgefertigte Pakete, die Ihnen die Einrichtung gemäß eines spezifischen Zwecks erleichtern, indem benötigte Pakete gruppiert werden. Diese Pakete tragen die Bezeichnung *task-* am Anfang des Paketnamens (siehe Abschnitt 2.7). Dazu zählen bspw. die Aufgaben Mailserver, Webserver, Desktopumgebung und Laptop (siehe Abbildung 6.3).

---

tasksel und andere Programme

Wenn das Paket `tasksel` installiert ist, zeigen sowohl Aptitude wie auch Synaptic (siehe Abschnitt 6.4.1) ebenfalls alle verfügbaren Aufgaben an. Aptitude verwendet dafür einen eigenen Ast als Sektion „Debian“ und Distributionsbereich „Tasks“, bei Synaptic hingegen heißt der Bereich (Sektion) „Tasks“.

Die textbasierte Benutzeroberfläche von `tasksel` ist jedoch nur eine Seite der Medaille. Das Programm ist ebenso für eine Steuerung über die Kommandozeile empfänglich. Die nachfolgende Liste zeigt die möglichen Schalter:

`install Aufgabe`
: installiert alle Pakete, die für die *Aufgabe* notwendig sind

`remove Aufgabe`
: entfernt alle Pakete, die zur angegebenen *Aufgabe* gehören

`--list-tasks`
: listet alle Aufgaben auf, die `tasksel` kennt

`--task-desc Aufgabe`
: zeigt eine Beschreibung der gewählten *Aufgabe* an

`--task-packages Aufgabe`
: zeigt alle Pakete an, die zur gewählten *Aufgabe* gehören

`-t` (Langform `--test`)
: Trockendurchlauf, Ausführung der gewünschten Aktion ohne echte Auswirkung

Über den Schalter `--list-tasks` stellt Ihnen `tasksel` alle vorab definierten Aufgaben zusammen (Debian). Am Buchstaben in der ersten Spalte der Ausgabe erkennen Sie, ob diese Aufgabe vollständig auf ihrem Linuxsystem umgesetzt wurde. Daneben sehen Sie das vergebene Kürzel und eine Kurzbeschreibung zur jeweiligen Aufgabe.

Ausgabe aller festgelegten Aufgaben von tasksel

```
$ tasksel --list-tasks
u desktop          Debian desktop environment
u web-server       Web server
u print-server     Printserver
u database-server  SQL database
u dns-server       DNS Server
u file-server      File server
u mail-server      Mail server
u ssh-server       SSH server
u laptop           Laptop
$
```

Für jede Aufgabe ist eine Beschreibung der Aufgabe hinterlegt. Diese zeigen Sie mit dem Schalter `--task-desc` an[3]. Auf einem Ubuntu mit `tasksel` in der Version 2.88 sehen Sie diese Ausgabe:

Ausgabe der Aufgabenbeschreibung eines tasks (Ubuntu)

[3] Unter Debian 7 *Wheezy* ist die Ausgabe derzeit defekt und als Bug #756841 hinterlegt, siehe https://bugs.debian.org/756841

```
$ tasksel --task-desc openssh-server
Selects packages needed for an Openssh server.
$
```

`tasksel` zeigt Ihnen mit Hilfe des Schalters `--task-packages` auch die Pakete an, die zu der entsprechenden Aufgabe gehören. Bei Debian und der Aufgabe *ssh-server* sieht das wie folgt aus — es verweist auf ein entsprechendes Debianpaket:

Pakete, die zu einer Aufgabe gehören (Debian)

```
$ tasksel --task-packages ssh-server
task-ssh-server
$
```

Der gleiche Aufruf auf einem Ubuntu — hier für das Paket *openssh-server* — ergibt diese Liste (Auszug) mit allen benötigten Einzelpaketen:

Pakete, die zu einer Aufgabe gehören (Ubuntu)

```
$ tasksel --task-packages openssh-server
python-six
python-chardet
python2.7
tcpd
openssh-server
ncurses-term
ssh-import-id
...
$
```

6.3.2 `aptitude`

Im Vergleich mit den anderen vorgestellten Programmen zur Paketverwaltung ist `aptitude` eine recht komplexe und umfangreiche Anwendung. Es ermöglicht Ihnen zwei unterschiedliche Wege der Bedienung — einerseits über die Kommandozeile mit Unterkommandos und Schaltern, andererseits über eine Ncurses-basierte, interaktive, farbige Bedienoberfläche im Terminal. Wieder aufgegeben wurden zwischenzeitlich die Versuche, `aptitude` auch mit einer graphischen Bedienoberfläche auszustatten (siehe [Beckert-Blog-Aptitude-Gtk-Will-Vanish]).

Das Programm ist verteilt auf die beiden Pakete namens *aptitude* und *aptitude-common*. Da das Programm nicht zur Standardauswahl bei der Installation von Debian GNU/Linux und Ubuntu gehört, richten Sie es am besten über den Aufruf `apt-get install aptitude` auf ihrem Linuxsystem ein. Das Paket `aptitude-common` wird über Paketabhängigkeiten automatisch mitinstalliert.

Ähnlich wie APT arbeitet `aptitude` mit Paketen, die sich entweder bereits lokal auf ihrem System befinden, oder noch auf einem Paketmirror vorliegen und vor der Installation noch von dort bezogen werden. Desweiteren bietet Ihnen das Programm die folgenden Funktionen (Auswahl, jeweils Angabe der Unterkommandos auf der Kommandozeile):

- die Liste der installierten Pakete anzeigen und ausgeben (Abschnitt 8.5) mit `aptitude search ~i`
- Recherche und Paketliste filtern anhand von Paketkategorien, Veröffentlichungen und Mustern (Teilzeichenketten, Reguläre Ausdrücke) bzgl. des Paketnamens, der Metadaten und der Paketbeschreibung
- Paketstatus erfragen (Abschnitt 8.4) mit `aptitude show Paketname`
- Paketdatei ins aktuelle Verzeichnis herunterladen (Abschnitt 8.31) mit `aptitude download Paketname`
- Pakete zur Installation, Aktualisierung oder Löschung vormerken (siehe Mit `aptitude` Vormerkungen machen unter Kapitel 11)
- Paket installieren (Abschnitt 8.36) mit `aptitude install Paketname`

- Paket in einer bestimmten Version halten (nicht aktualisieren) (siehe Kapitel 15 und Kapitel 16)
- Pakete deinstallieren (Abschnitt 8.41) mit `aptitude remove Paketname`
- Pakete erneut installieren (Abschnitt 8.37) mit `aptitude reinstall Paketname`
- installierte Paketliste oder die Veröffentlichung aktualisieren (Abschnitt 8.39) mit `aptitude update`, `aptitude safe-upgrade` und `aptitude full-upgrade`
- klären, warum ein Paket (nicht) installiert ist (Abschnitt 8.15) mit `aptitude why Paketname` bzw. `aptitude why-not Paketname`
- Paketabhängigkeiten mit `aptitude search ?depends Paketname` anzeigen (Abschnitt 8.17)

Dokumentation zu aptitude
Für den vollständigen Funktionsumfang und als Einstieg zum Programm ist das Lesen der Dokumentation zu `aptitude` ([aptitude-dokumentation]) empfehlenswert. Neben der Bedienung enthält es alle Unterkommandos, Optionen, Schalter und Möglichkeiten zur Konfiguration.

Wie bereits oben angerissen, können Sie `aptitude` über die **Kommandozeile** benutzen. Die Unterkommandos und Schalter sind bzgl. der Schreibweise und Bedeutung ähnlich derer von APT (siehe Abschnitt 6.2.2).

Um hingegen über die **Ncurses-basierte Bedienoberfläche** zu agieren, starten Sie zunächst `aptitude` ohne weitere Optionen. Die mehrfarbige Bedienoberfläche enthält mehrere Elemente. Ganz oben finden Sie die verfügbaren, aktiven Tasten und deren Funktion. Über die Funktionstaste **F10** oder alternativ mit Hilfe der Tastenkombination Ctrl-T aktivieren Sie bspw. die Menüleiste. Einige Terminals wie bspw. das Gnome-Terminal fangen die Funktionstaste ab und belegen diese für die eigene Menüleiste. Über den Eintrag Bearbeiten → Tastenkombinationen → Menütastenkombinationen aktivieren (de)aktivieren Sie das Verhalten. Mit Hilfe der Pfeiltasten navigieren Sie zwischen den einzelnen Menüeinträgen hin und her bzw. wählen die gewünschte Aktion aus.

Die beiden Fensterhälften darunter geben Ihnen eine Übersicht zu den Softwarepaketen. In der oberen Hälfte stellt `aptitude` in einer aufklappbaren Baumstruktur die Paketkategorien (Abschnitt 2.8), den Distributionsbereich (Abschnitt 2.9) und den Paketnamen mit Versionsnummer dar. Sichtbar wird dabei die Version des installierten Pakets sowie der möglichen Aktualisierung (Abschnitt 2.11.2). Die Auswahl in der Baumstruktur erfolgt analog zu vi(m) mittels **j** und **k** (oder über die Pfeiltasten) und **Enter**. Die einzelnen Strukturebenen klappen Sie mit den Tasten **Enter**, **[** und **"**"] auf und zu.

Dabei hinterlegt `aptitude` die einzelnen Pakete mit verschiedenen Farben, deren Bedeutung Sie Tabelle 6.2 entnehmen. Das Farbschema können Sie auch nach Gutdünken anpassen, genauer gehen wir darauf in Abschnitt 10.8 ein.

Tabelle 6.2: Farben und deren Bedeutung bei `aptitude`

Farbkombination	**Bedeutung**
schwarzer Hintergrund mit weißer Schrift	das Paket wird nicht verändert
roter Hintergrund mit weißer Schrift	das Paket ist defekt oder kann nicht installiert werden
blauer Hintergrund mit weißer Schrift	das Paket wird aktualisiert
weißer Hintergrund mit schwarzer Schrift	die Paketversion bleibt erhalten, kann jedoch aktualisiert werden
grüner Hintergrund mit schwarzer Schrift	Paket wird installiert
lila Hintergrund mit schwarzer Schrift	Paket wird entfernt („deinstalliert“)

Im unteren Fenster erhalten Sie eine Beschreibung — entweder zur ausgewählten Paketkategorie oder zum jeweiligen Paket. Zwischen den beiden Fensterhälften wechseln Sie mittels der **Tab**-Taste hin und her. Die Belegung weiterer Tasten entnehmen Sie bitte Tabelle 6.3.

Tabelle 6.3: Tasten bei `aptitude`

Aktion	Tastenbelegung
Hilfe	**?**
`aptitude` beenden (Vormerkungen werden gespeichert)	Shift-q
`aptitude` abbrechen (alle Vormerkungen gehen verloren)	Ctrl-c
Info-Fenster ein- und ausblenden	Shift-d
Zwischen den Info-Ansichten wechseln	**i**
Zwischen beiden Fenstern hin- und herwechseln	**Tab**
In das Menü von `aptitude` wechseln	Ctrl-t oder **F10**
Paketlisten aktualisieren	**u**
Ausgewähltes Paket auswählen	-
Ausgewähltes Paket entfernen	**-**

Auch wenn sich APT und `aptitude` größtenteils sehr ähnlich sind, bestehen eine Reihe von feinen Unterschieden, die erst während der Benutzung der Programme präsent werden. `aptitude` hat nützliche Erweiterungen, wie z.B. einen **interaktiven Abhängigkeitsauflöser** (siehe Abbildung 6.4). Verändern Sie den geplanten Paketbestand, indem Sie beispielsweise ein zusätzliches Paket markieren und somit zur Installation vormerken, werden automatisch die notwendigen Abhängigkeiten aufgelöst und ebenfalls vorgemerkt. Sie sehen somit unmittelbar, welche Pakete im nächsten Schritt noch hinzukommen oder wieder entfernt werden müssen.

Sollte es dabei zu Paketkonflikten kommen, so werden Ihnen vorab verschiedene Lösungsvarianten und deren Auswirkungen auf den Paketbestand zur Auswahl gestellt. Im Gegensatz dazu präsentiert Ihnen APT nur stets einen einzigen Vorschlag zur Aktualisierung.

Aus diesen angebotenen Varianten wählen Sie die Ihnen am besten passende aus. In den letzten beiden Zeilen des Terminals listet `aptitude` auf, wieviele Varianten es ermittelt hat, mit welchen Tasten Sie zwischen diesen Varianten wechseln (siehe auch Tabelle 6.4) und wie Sie die gewünschte Variante letztendlich auswählen.

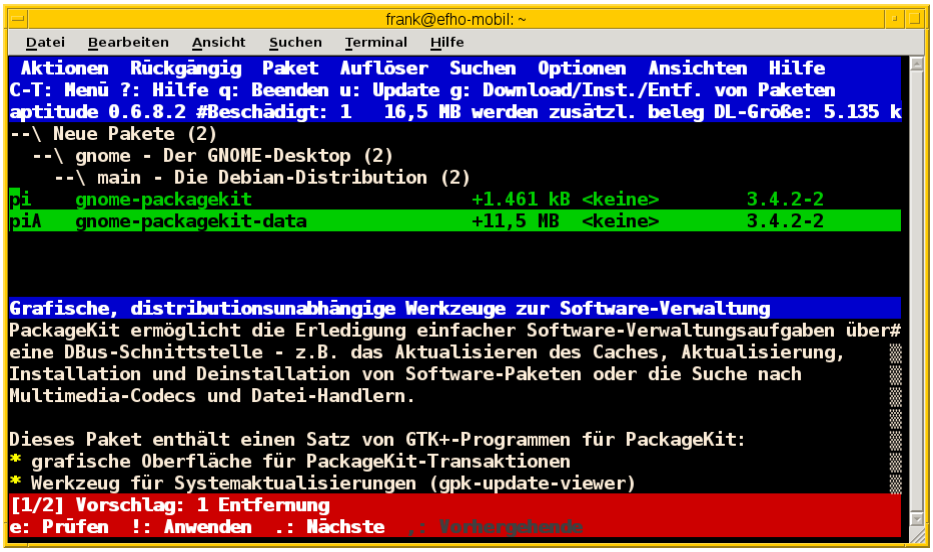

Abbildung 6.4: package dependency solver in `aptitude`

Tabelle 6.4: Tasten zur interaktiven Konfliktlösung bei `aptitude`

Aktion	Tastenbelegung
Vorschläge zur Konfliktlösung anzeigen	**e**
Nächsten Vorschlag anzeigen	**.**

Tabelle 6.4: (continued)

Aktion	Tastenbelegung
Vorherigen Vorschlag anzeigen	,
Ersten Vorschlag anzeigen	<
Letzten Vorschlag anzeigen	>
Teilvorschlag akzeptieren	a
Teilvorschlag ablehnen („reject“)	r
Vorschlag anwenden	!

Darüber hinaus verfügt `aptitude` über eine Ansicht, in der Sie Pakete nach **Debian-Tags (Debtags)** (siehe dazu Kapitel 13) sortiert betrachten können. Damit stöbern Sie sehr effizient im Paketbestand. Das ist insbesondere dann interessant, wenn Sie lediglich wissen, nach welcher Funktionalität oder Art von Paket Sie suchen, jedoch den konkreten Paketnamen nicht kennen.

Der ebenfalls im Menü in Abbildung 13.5 (noch) angezeigte **Kategoriebrowser** gilt als veraltet[4], funktioniert seit einigen Versionen nicht mehr und wird voraussichtlich demnächst ganz entfernt ([aptitude-categorical-browser-to-be-removed]). Der oben angerissene Debtags-Browser ist der offizielle, wesentlich aktuellere und besser gepflegte Ersatz dafür.

Im Erweiterungsteil gehen wir darauf ein, was passiert, wenn Sie APT und `aptitude` miteinander mischen (Kapitel 12). Auch der Konfiguration des Programms ist ein eigener Abschnitt gewidmet (siehe APT und `aptitude` auf die eigenen Bedürfnisse anpassen in Kapitel 10).

6.4 GUI zur Paketverwaltung

6.4.1 Synaptic

Das Programm steht im gleichnamigen Paket *synaptic* [Debian-Paket-synaptic] bereit. Es verfügt über eine graphische Bedienoberfläche auf der Basis des Gimp Toolkits (GTK2) und war lange Zeit das empfohlene Programm zur Paketverwaltung für die Benutzer des Ubuntu-Desktops. Inzwischen hat das mehr an Apples App-Store denn an Aptitude erinnernde Ubuntu Software Center (Abschnitt 6.4.4) viele Fans abgeworben.

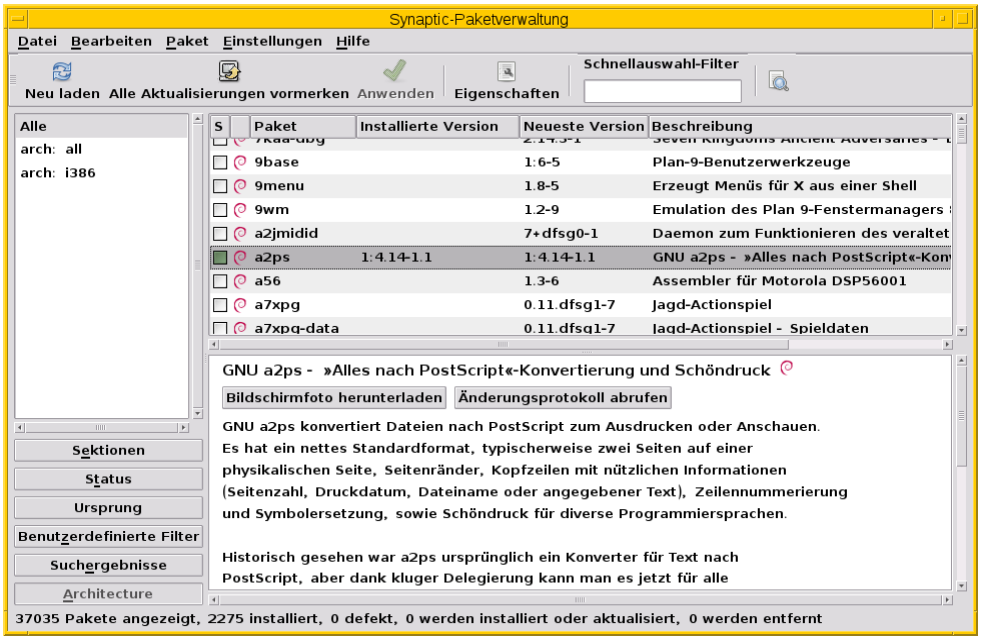

Abbildung 6.5: Softwareauswahl in `synaptic`

[4] Es handelt sich dabei um eine hart in `aptitude` verdrahtete und schon sehr lange nicht mehr gepflegte Kategorisierung der Pakete

Synaptic bedienen Sie über die Menüleiste, eine Reihe von Knöpfen darunter und eine dreispaltige Paketübersicht. Die Darstellung konfigurieren Sie über die Menüpunkte Einstellungen → Einstellungen und Einstellungen → Werkzeugleiste. Abbildung 6.6 zeigt als Beispiel das Dialogfenster, über welches Sie die Farben zum jeweiligen Installationsstatus eines Pakets festlegen.

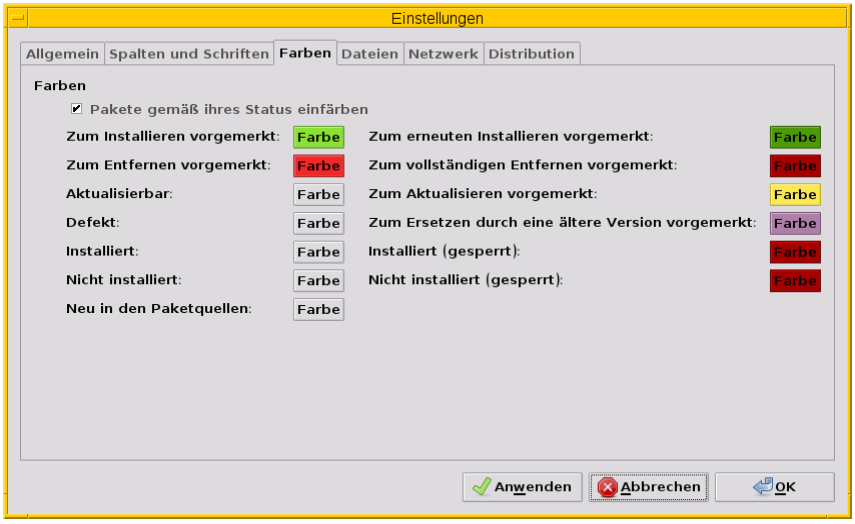

Abbildung 6.6: Farbige Markierungen der Pakete gemäß ihrem Installationsstatus (Synaptic)

Über den Knopf Eigenschaften erfahren Sie mehr über das gerade von Ihnen ausgewählte Paket. Dazu zählen Allgemeine Informationen, die Paketabhängigkeiten, die installierten Dateien, die verfügbaren Paketversionen sowie eine ausführliche Paketbeschreibung. Abbildung 6.7 zeigt die Informationen zum Paket *ding*.

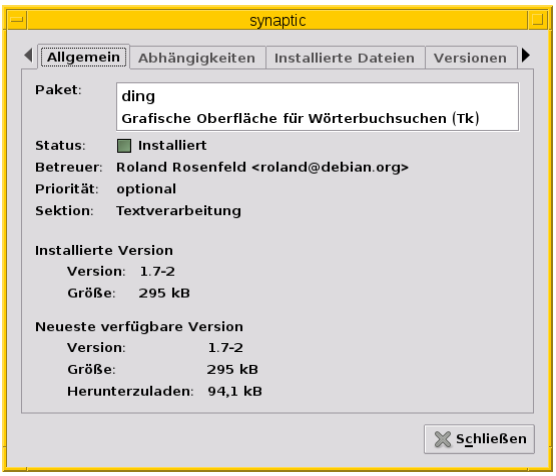

Abbildung 6.7: Allgemeine Paketeigenschaften für das Paket *ding* (Synaptic)

Unter der Menüleiste und den Knöpfen finden Sie die dreispaltige *Paketübersicht*. Links finden Sie verschiedene Auswahlknöpfe, oben rechts die Paketliste und unten rechts die Paketbeschreibung im Detail. Abbildung 6.5 zeigt Ihnen die Gesamtansicht anhand des Pakets *a2ps*.

Die *linke Spalte* zeigt zunächst die Architektur (Abschnitt 1.2). Über die einzelnen Knöpfe darunter schalten Sie zur Ansicht nach den Paketkategorien (Sektionen) (Abschnitt 2.8) sowie dem Ursprung bzw. der Herkunft der Pakete (Abschnitt 3.1), der Veröffentlichung (Abschnitt 2.10) und dem Distributionsbereich (Abschnitt 2.9) um.

In der *Paketliste oben rechts* beinhalten die Spalten den Installationsstatus (Status), eine Information zur Herkunft des Pakets, den Paketnamen, die installierte und die verfügbare Version und eine kurze Paketbeschreibung. Zusätzlich können Sie als Spalten

den Distributionsbereich, die Veröffentlichung und die Größe des Pakets nach der Installation ergänzen. Mit einem Mausklick auf den jeweiligen Spaltenkopf sortieren Sie die Paketliste nach der jeweiligen Eigenschaft.

Die *rechte untere Spalte* zeigt die ausführliche Paketbeschreibung an. Über den linken Knopf (Bildschirmfoto herunterladen) beziehen Sie ein Bildschirmfoto, sofern dieses hinterlegt ist[5]. Über den rechten Knopf (Änderungsprotokoll abrufen) zeigt Ihnen Synaptic die Änderungsdatei (engl. *Changelog*) zum ausgewählten Paket an.

Um ein Paket zu installieren, wählen Sie dieses zuerst über den Menüeintrag Paket → Zum installieren vormerken (alternativ Strg-I oder einen Rechtsklick) aus. Über den Menüeintrag Bearbeiten → Vorgemerkte Änderungen anwenden (alternativ Strg-P oder den Knopf Anwenden) lösen Sie die Installation aus. In ähnlicher Art und Weise verfahren Sie beim Löschen und Aktualisieren von Paketen. Synaptic prüft bei jeder Aktion die Paketabhängigkeiten und bezieht die weiteren Pakete in die Verarbeitung mit ein, damit ihr Linuxsystem stets in einem konsistenten Zustand bleibt.

Möchten Sie hingegen eine ganze Paketgruppe installieren, bietet Synaptic die gleiche Funktionalität wie das Werkzeug *tasksel* (siehe Abschnitt 6.3.1). Dazu nutzen Sie den Menüpunkt Bearbeiten → Pakete nach Aufgaben vormerken. Daraufhin erscheint ein ähnliches Auswahlfenster wie in Abbildung 6.8, aus deren Liste sie die gewünschte Aktion markieren. Alle Pakete, die der ausgewählten Aufgabe zugeordnet sind, gelangen damit in die Vorauswahl und können daraufhin über den Knopf Anwenden installiert werden.

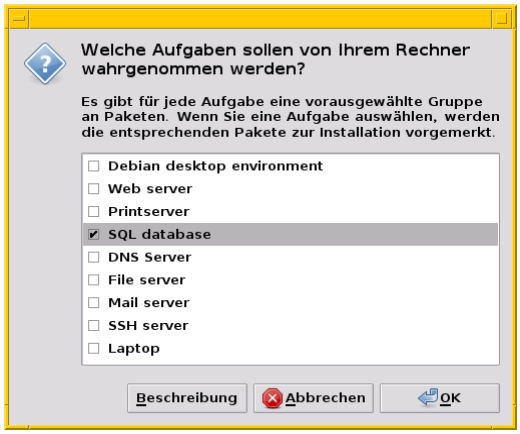

Abbildung 6.8: Paketauswahl einer ganzen Aufgabengruppe (Synaptic)

6.4.2 Muon

* Klon von Synaptic (siehe Abschnitt 6.4.1) * setzt auf GTK auf * bislang speziell genutzt in Kubuntu * Debianpaket *muon* [Debian-Paket-muon]

6.4.3 Smart Package Management (SmartPM)

Im Paket *smartpm* [Debian-Paket-smartpm] verbirgt sich das gleichnamige Programm zur Paketverwaltung. Zunächst etwas unscheinbar, entpuppt es sich aber bei näherer Betrachtung als eine Art Alleskönner und mindestens gleichwertiges Pendant zu Synaptic (siehe Abschnitt 6.4.1).

SmartPM verfügt über drei Bedienmodi. Erstens hat es ebenfalls eine graphische Bedienoberfläche auf der Basis des Gimp Toolkits (GTK2), lässt sich jedoch zweitens auch über die Kommandozeile mit mehreren Schaltern steuern und verfügt als drittes noch über eine Paketverwaltungsshell analog zur *aptsh* (Abschnitt 6.2.3), zu *wajig* (Abschnitt 8.42.7) und zu *cupt* (Abschnitt 6.2.5).

Für ersteres rufen Sie SmartPM im Terminal über das Kommando `smart --gui` auf oder wählen den entsprechenden Eintrag aus dem Menü Ihrer Desktop-Umgebung aus. In Abbildung 6.9 sehen Sie die zweispaltige Darstellung — links die Paketkategorien, rechts oben die Paketliste mit Paketname samt Versionsnummer und rechts unten die ausführliche Paketbeschreibung — hier

[5] Die Bildschirmfotos kommen von [screenshots.debian.net]. Falls für Ihr Lieblingspaket ein Screenshot fehlt, können Sie selbst einen anfertigen und dort hochladen. Nach einem Review wird das hochgeladene Bild im Normalfall freigeschaltet und ist dann für alle Nutzer der Webseite und der Programme, die die Daten von dort verwenden, sichtbar.

am Beispiel des Pakets *kexi*. Unter dem Reiter General verbergen sich die Basisinformationen zum Paket, Description bietet die Paketbeschreibung, Content die Dateien aus dem Paket, Changelog die Veränderungen zur vorherigen Version, Relations die darüber bereitgestellten, verfügbaren und zusätzlich benötigten Pakete. Unter dem Reiter URLs verbergen sich weitere Referenzen zum Paket.

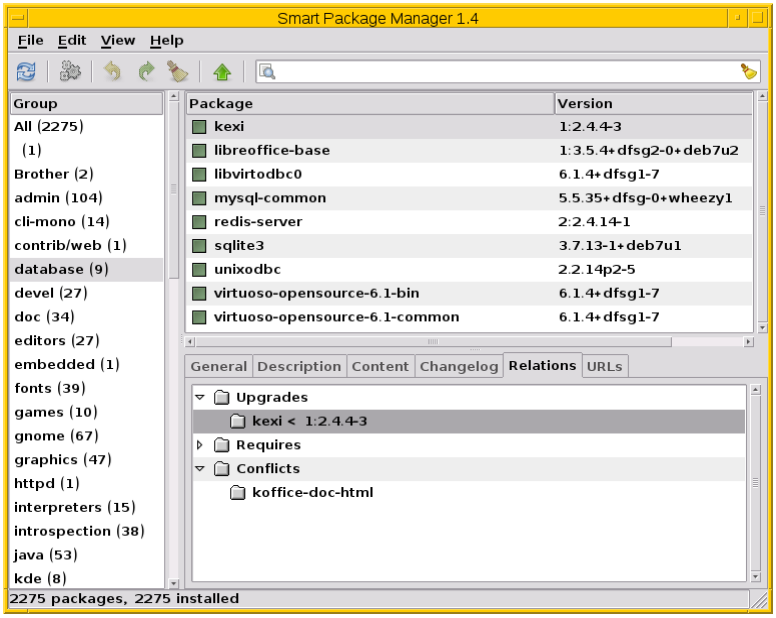

Abbildung 6.9: Softwareauswahl in `smartpm`

Für die Benutzung von SmartPM über die *Kommandozeile* starten Sie das Programm über den Aufruf `smart` mit der gewünschten Aktion und dem Paketname. Analog zu APT bzw. `aptitude` stehen bspw. die Unterkommandos `install`, `remove` und `upgrade` bereit.

Über den Aufruf `smart --shell` erreichen Sie die Paketverwaltungsshell. Darin nutzen Sie die gleichen Unterkommandos und Schalter wie bei obigem Aufruf über die Kommandozeile. Nachfolgendes Beispiel zeigt das Unterkommando `info`, welches hier alle Paketinformationen zum Paket *kexi* ausgibt.

Paketinformationen zu kexi in der Shell von SmartPM

```
$ smart --shell
Smart Package Manager 1.4 - Shell Mode

Loading cache...
Updating cache...               ######################################## [100%]

smart> info kexi
Name: kexi
Version: 1:2.4.4-3
Priority: 0
Source: calligra_1:2.4.4-3
Group: database
License:
Installed Size: 8.8MB
Reference URLs: http://www.calligra-suite.org/kexi/
Flags:
Channels: DEB System
Summary: integrated database environment for the Calligra Suite
Description:
 Kexi is an integrated data management application. It can be used for
 creating database schemas, inserting data, performing queries, and
 processing data. Forms can be created to provide a custom interface to
```

```
 your data. All database objects - tables, queries and forms - are stored
 in the database, making it easy to share data and design.
 .
 Kexi is considered as a long awaited Open Source competitor for MS Access,
 Filemaker and Oracle Forms. Its development is motivated by the lack of
 Rapid Application Development (RAD) tools for database systems that are
 sufficiently powerful, inexpensive, open standards driven and portable
 across many operating systems and hardware platforms.
 .
 This package is part of the Calligra Suite.

smart>
```

SmartPM wirkt sehr ausgereift und verfügt zudem über eine Reihe von *Besonderheiten*. Es kann sowohl mit Paketen im `deb`- als auch in den verschiedenen `rpm`-Formaten umgehen. Das kann recht praktisch in gemischten Umgebungen sein. Im Gegensatz zu APT und `aptitude` gestattet es die Auswahl einer oder mehrerer Paketquellen zur Aktualisierung — bei APT sind nur alle aktiven auf einmal möglich.

Analog zu APT und `aptitude` kennt SmartPM auch diverse Markierungen. Das sind beispielsweise Flags, die anzeigen lassen, ob ein Paket seit der letzten Aktualisierung der Paketlisten neu hinzukam, ob ein Paket nicht aktualisiert werden darf („lock“, „hold“), oder ob ein Paket automatisch installiert wurde[6].

Zusätzlicher Lesestoff
Eine ausführliche Beschreibung zum Programm mit weiteren Beispielen zur Konfiguration und zur Handhabung entnehmen Sie bitte dem Linux-User-Artikel zum gleichen Thema [Hofmann-Smartpm-LinuxUser].

6.4.4 Ubuntu Software Center

Canonical bietet mit dem Ubuntu Software Center [Ubuntu-Software-Center] eine weitere graphische Verwaltung zur Auswahl und Pflege ihres Softwarebestands an. Dieses Paket wurde 2010 auch in den Bestand von Debian als Paket *software-center* [Debian-Paket-software-center] übernommen, aber bereits 2014 wieder aus Debian entfernt. Die Begründung ist, dass das Programm nicht mehr aktuell in Debian sei und auch bessere Alternativen existieren ([RM-software-center]). Es ist daher in Debian 6 *Squeeze* und Debian 7 *Wheezy* enthalten, aber nicht mehr ab Debian 8 *Jessie*. Für Ubuntu steht es weiterhin stets in aktualisierter Form zur Verfügung [Ubuntu-Paket-software-center].

Wie bei den anderen bereits weiter oben vorgestellten Programmen können Sie damit nicht nur die benutzten Paketquellen verwalten, d.h. hinzufügen, ändern, löschen und verifizieren, sondern auch den Softwarebestand ihrer Systeme verändern. Über die Benutzeroberfläche erhalten Sie eine Anwendungsübersicht nach Kategorien, wählen daraus die gewünschten Softwarepakete aus und installieren, entfernen und aktualisieren diese (siehe Abbildung 6.10).

[6] Bislang scheint SmartPM diese Markierungen nicht mit APT oder `aptitude` zu synchronisieren. Dieses Verhalten ist als Bug registriert.

Abbildung 6.10: Softwareauswahl im Ubuntu Software Center (aus Xubuntu 13.04)

Bis zur Jahresmitte 2014 bestand seitens Canonical der Datenaustauschdienst Ubuntu One (siehe dazu [Ubuntu-One] und [Ubuntu-One-Wikipedia]). Das Ubuntu Software Center war in diesen Dienst integriert und bot Ihnen die Möglichkeit, über das Programm den Softwarebestand zwischen verschiedenen Rechnern abzugleichen. Das beinhaltete ebenso die Veränderungen im Paketbestand und listete auf, welche Pakete wann installiert, aktualisiert oder gelöscht wurden (siehe Abbildung 6.11). Wie Ihnen das gleiche auf der Kommandozeile gelingt, lesen Sie unter Liste der zuletzt installierten Pakete anzeigen in Abschnitt 8.16.

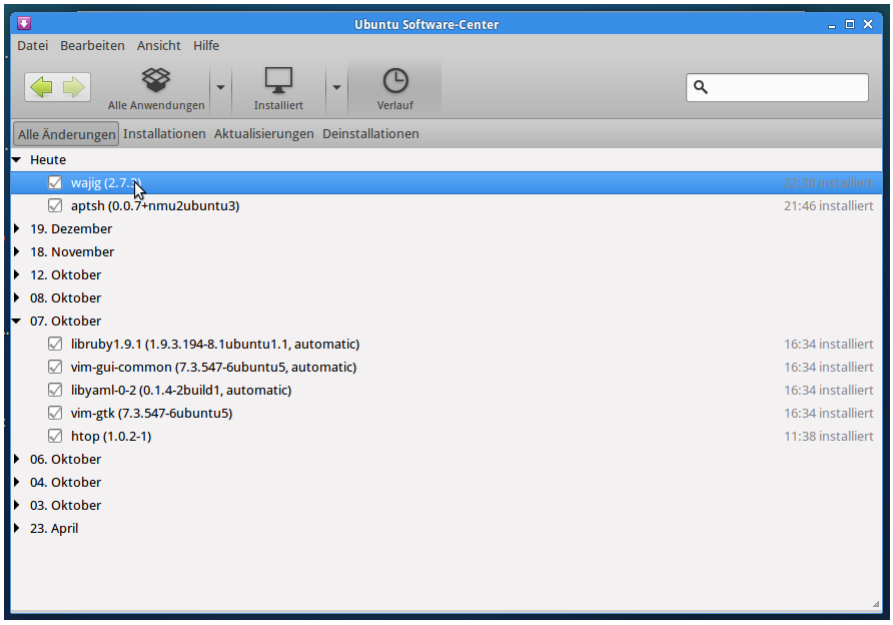

Abbildung 6.11: Verlauf der Änderungen im Paketbestand

Bei der Benutzung des Programms beachten Sie bitte, dass das Programm viele graphische Elemente und Inhalte enthält, die über das Internet bereitgestellt werden. Daher empfehlen wir Ihnen zur Benutzung eine Internetverbindung, da ansonsten viele Informationen in der Bedienoberfläche nicht angezeigt werden können. Desweiteren ist das Programm nur weitestgehend mit der Maus bedienbar und kaum über die Tastatur. Das ist nicht für alle Nutzungskonstellationen und Anwender hilfreich.

6.4.5 PackageKit

PackageKit ist eine allgemeine, distributionsneutrale Schnittstelle für unterschiedliche Paketverwaltungen, eine sogenannte Abstraktionsebene für die Paketverwaltung (*package management abstraction layer*). Das Designziel besteht darin, alle graphischen Werkzeuge zu vereinigen, die bei den verschiedenen Linuxdistributionen im Einsatz sind und gleichzeitig auf die neueste Technologie wie PolicyKit[7] umzustellen. PackageKit ist nicht dafür gedacht, hochspezialisierte Paketverwaltungssoftware zu ersetzen.

Seit 2009 nutzt die Linuxdistribution Kubuntu diese Schnittstelle für seine Paketverwaltung. Weitere Anwendungen, die auf PackageKit aufsetzen, sind z.B. das Paket *apper* [Debian-Paket-apper] für den KDE, das Paket *gnome-packagekit* [Debian-Paket-gnome-packagekit] für GNOME (siehe Abbildung 6.12) und das Installationsprogramm zu Openmoko [OpenMoko].

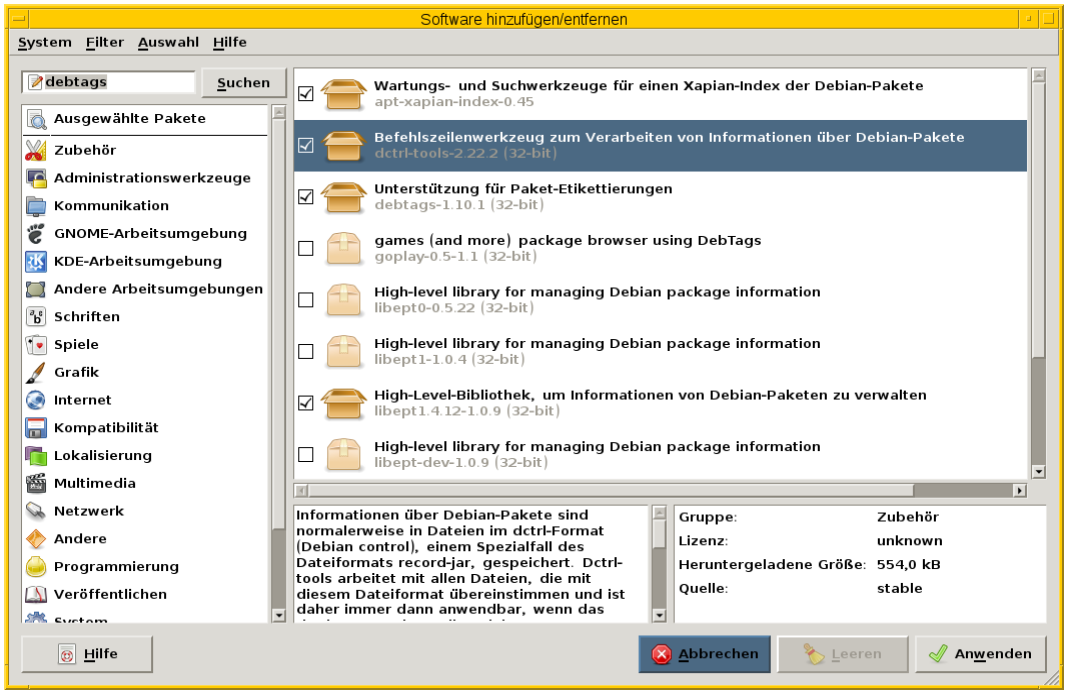

Abbildung 6.12: Gnome Packagekit mit ausgewähltem Paket *dctrl-tools*

PackageKit kann über die Interfaces APT und `aptcc` (Paket *packagekit-backend-aptcc* [Debian-Paket-packagekit-backend-aptcc]) an APT andocken. Ähnliches liefert das Paket *packagekit-backend-smart* [Debian-Paket-packagekit-backend-smart] für die Zusammenarbeit mit SmartPM (Abschnitt 6.4.3).

6.4.6 GDebi

Initial nur für Ubuntu entwickelt, verbirgt sich unter dem Namen GDebi [gdebi] ein kleines, seit 2004 von Michael Vogt betreutes Programm. Auf den ersten Blick wirkt GDebi recht unscheinbar, füllt aber genau die kleine Lücke zwischen den Kommandos `dpkg -i` und `apt-get install` (siehe Abschnitt 6.2.1, Abschnitt 6.2.2 und Abschnitt 8.36). Die Besonderheit von GDebi liegt darin, dass es lokal vorliegende `deb`-Pakete installieren kann, während es deren Abhängigkeiten aus den Repositories via APT auflöst und daraus die zusätzlich benötigten Pakete bezieht. Damit eignet es sich besonders gut, um zugesandte, selbst erstellte oder auch manuell heruntergeladene Pakete zu installieren.

GDebi besteht aus drei **Komponenten** — einem Kommandozeilenprogramm namens `gdebi` (aus dem Paket *gdebi-core* [Debian-Paket-gdebi-core]) und zwei graphischen Bedienoberflächen. Es existiert eine GTK-basierte Variante namens `gdebi-gtk` (aus dem Paket *gdebi* [Debian-Paket-gdebi] — siehe Abbildung 6.13) und eine KDE-Variante namens `gdebi-kde` (aus dem gleichnamigen Paket *gdebi-kde* [Debian-Paket-gdebi-kde]).

[7] Berechtigungsdienst, der die Kommunikation von Software via DBus-Protokoll untereinander regelt

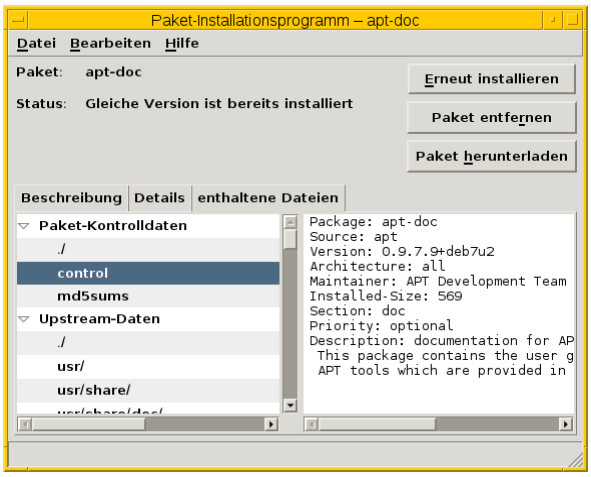

Abbildung 6.13: `gdebi` mit den Informationen zum Paket *apt-doc*

Dabei dient das Kommandozeilenprogramm `gdebi` den beiden graphischen Werkzeugen als Backend und wird von diesen intern aufgerufen, um die jeweiligen Paketoperationen auszuführen. Sie können es aber auch alleine auf der Kommandozeile benutzen, ohne dass eine der beiden graphischen Komponenten installiert sein muss. Rufen Sie dazu GDebi auf der **Kommandozeile** mit einem `deb`-Paket als Parameter auf, erhalten Sie die Paketbeschreibung. Stimmen Sie danach der abschließenden Frage zur Installation mit **j** zu, führt `gdebi` ihren Wunsch aus und das `deb`-Paket landet auf ihrem Linuxsystem.

Anzeige der Paketbeschreibung bei manuellem Aufruf von gdebi

```
# gdebi Desktop/odeskteam_3.10.5_debian_7.2_i386.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Building data structures... Done
Building data structures... Done


oDesk Team - complete time-logging and verification system
 Who needs it: All providers are required to run oDesk Team in order to have a
 verified record of their work. oDesk Team is optional for buyers, however a
 lot of our buyers run it to have an online record of their work and be able to
 collaborate better with their remote team.
 .
 Note: This single-install download is the full oDesk Team client application,
 which will only be fully functional if used in conjunction with an oDesk Team
 Online Account with a valid license to access the Service. Review carefully
 our License Agreement before downloading.
Wollen Sie das Software-Paket installieren? [j/N]:
...
#
```

Die **graphischen Werkzeuge** starten Sie entweder über den Aufruf `gdebi-gtk` bzw. `gdebi-kde` im Terminal, oder aber durch Doppelklicken auf eine `deb`-Datei im Dateimanager ihrer Desktop-Umgebung. In letzterem Fall wird das von Ihnen angeklickte Paket jedoch nicht sofort installiert. Sie erhalten zunächst ein Fenster mit vielfältigen Informationen über das ausgewählte Paket (analog zu Abbildung 6.13). Erst mit einem weiteren Klick auf Paket installieren lösen Sie die Installation tatsächlich aus. Dieses Vorgehen hilft Ihnen dabei, herum(f)liegende Pakete nicht aus Versehen zu installieren. Somit vergewissern Sie sich nochmals, aus welcher ggf. auch unverifizierten Quelle dieses Paket stammt, bevor Sie es auf Ihrem Rechner installieren.

Neben der Installation und Aktualisierung ermöglicht Ihnen GDebi auch das Begutachten und Löschen von lokal vorliegenden Paketen. Dies können auch bereits vorher via APT heruntergeladene Pakete sein, die noch im Paketcache unter `/var/cache/apt/archives/` herumdümpeln (siehe Kapitel 7).

Das Begutachten von Paketen gelingt Ihnen über die einzelnen Reiter Beschreibung, Details und enthaltene Dateien. Neben der Paketbeschreibung zeigt Ihnen GDebi alle sonstigen Metadaten aus der Control-Datei (siehe Abschnitt 4.2) sowie den tatsächlichen Paketinhalt an, sofern es sich dabei um Textdateien handelt. Dies umfasst auch die Maintainer-Skripte.

Darüberhinaus zeigt Ihnen GDebi ab Version 0.9[8] auch sämtliche Warnungen des Programms `lintian` ([Debian-Paket-lintian]) an, die von dem ausgewählten Paket verursacht werden. Dies erlaubt Ihnen sowohl als Entwickler, als auch als normaler Benutzer, schnell einen groben Eindruck von der Qualität des Pakets zu bekommen, bevor Sie dieses auf ihrem Linux-System installieren und benutzen. Wie Sie mit dem referenzierten Programm `lintian` im Detail umgehen, lesen Sie unter Lintian in Abschnitt 32.1.

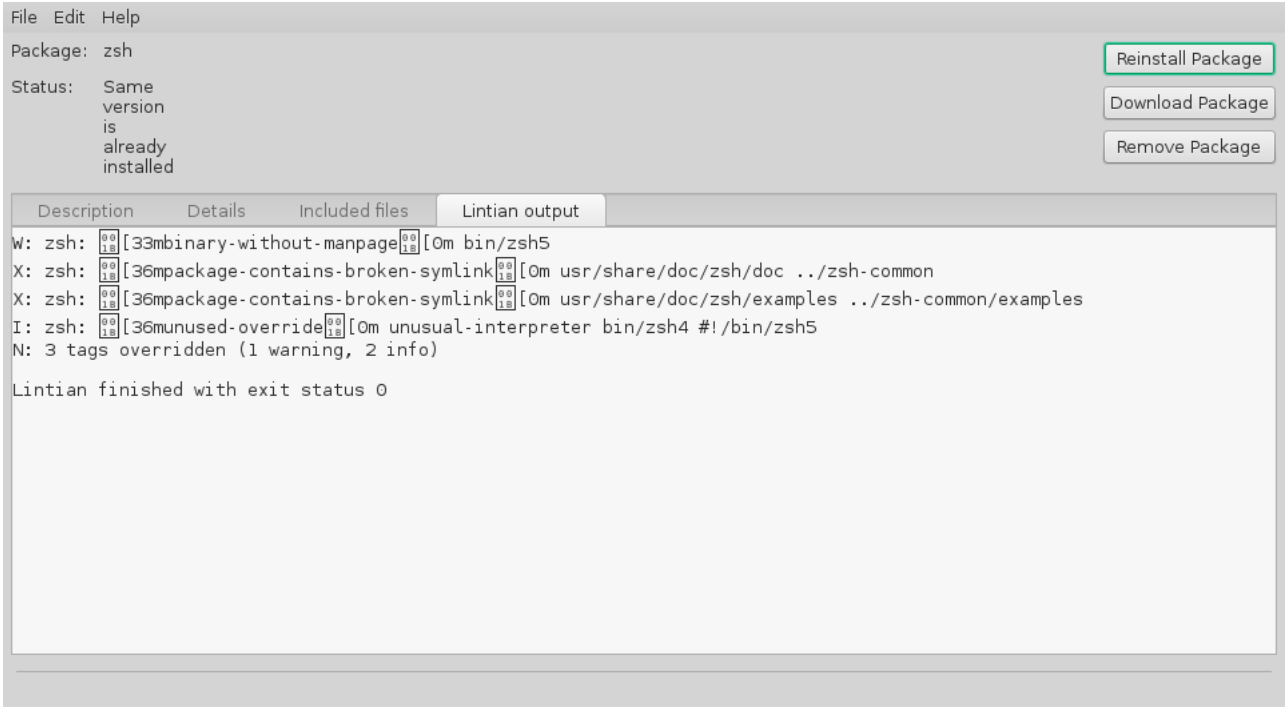

Abbildung 6.14: `gdebi-gtk` mit den Informationen zum Paket *zsh*

Ergibt sich bei der Veränderung des Paketbestands die Notwendigkeit, zusätzliche Paketabhängigkeiten aufzulösen, springt GDebi in die Bresche und klärt diese Situation automatisch mit Hilfe von APT ([Vogt-gdebi]). Fehlende Pakete werden von den vorab konfigurierten Paketmirrors (siehe Abschnitt 3.3) nachgezogen. Diese Eigenschaft hebt GDebi deutlich von `dpkg` ab, das nur meckern kann, falls es auf nicht-erfüllte Abhängigkeiten stößt.

Einziger Wermutstropfen bei GDebi ist, dass sowohl die beiden graphischen Tools, als auch `gdebi` bislang pro Aufruf nur ein einziges `deb`-Paket akzeptieren. APT ab Version 1.1 kann allerdings ebenfalls mit lokalen Paketen umgehen und dabei deren Abhängigkeiten über APT-Repositories auflösen — und das auch mit mehr als einem Paket auf einmal. Damit bietet es sich zukünftig als veritable Alternative zu `gdebi` an und soll dieses auch langfristig ersetzen[9].

6.5 Webbasierte Programme

6.5.1 In Paketen blättern mittels `dpkg-www`

Das Projekt `dpkg-www` umfasst ein CGI-Skript und stellt Ihnen damit den Zugang zu sämtlicher Dokumentation zu den einzelnen Paketen zur Verfügung, die auf ihrem eigenen oder einem fremden Debian-System installiert sind. Dazu analysiert es das Verzeichnis `/usr/share/doc`. Als Ergebnis erhalten Sie zu jedem Paket eine graphische Auflistung der dazugehörigen, verfügbaren Textdateien, README-Dokumente, Manpages und GNU Info-Dokumente. Auch das manuelle Durchblättern des

[8] Verfügbar ab Debian 8 *Jessie* und Ubuntu 14.04 LTS *Trusty Tahr*

[9] Letzteres ist auch kein Wunder, da sowohl `gdebi` als auch diese Funktionalität von APT vom gleichen Autor stammen.

Verzeichnisses `/usr/share/doc` gehört dazu. Ist ebenfalls das Paket *dctrl-tools* [Debian-Paket-dctrl-tools] installiert, können Sie mittels `dpkg-www` auch nach anderen Paketen und insbesondere nach den darin enthaltenen Dateien suchen.

Die Grundlage bildet das gleichnamige Paket *dpkg-www* ([Debian-Paket-dpkg-www]). Obwohl dieses seine letzte reguläre Aktualisierung bereits 2008 erhielt und derzeit als verwaist eingestuft ist (d.h, dass das Paket keinen Maintainer mehr hat), sind gemäß Popularity Contest ([Debian-Popularity-Contest]) noch etwa 350 Installationen in Benutzung (siehe dazu auch Abschnitt 2.14).

Bitte beachten Sie, dass Sie ohne Anpassung der Konfiguration lediglich im Paketbestand des Systems stöbern können. Aus der Bedienoberfläche heraus können Sie im Auslieferungszustand keine Veränderung ihres Paketbestands vornehmen. Auch die Autoren raten davon ab, da sie es als Sicherheitsrisiko einstufen. Um das dennoch zu ermöglichen, sind zunächst noch Änderungen an den beiden Konfigurationsdateien `/etc/dpkg-www.conf` (Apache) und `/etc/dwww/dwww.conf` (Webserver) erforderlich. Die Informationen dazu entnehmen Sie bitte der Dokumentation zu `dpkg-www`.

`dpkg-www` setzt auf einem installierten Webserver wie Apache oder Nginx auf. Daher ist die primäre Schnittstelle zur Bedienung des Programms auch ihr **Webbrowser**. Das Analyseergebnis stellt `dpkg-www` zweispaltig dar. Links stehen die verschiedenen Paketkategorien (siehe Abschnitt 2.8), rechts finden Sie die einzelnen Pakete, die der jeweiligen Kategorie zugeordnet sind (siehe Abbildung 6.15). Jedes Paket wird dabei mit seinem Titel, einer Kurzbeschreibung, dem dazugehörigen Paketnamen und mit der Angabe der bereitstehenden Formate der Dokumentation oder der weiterführenden Dokumente aufgelistet. In der Regel sind das HTML, Plaintext, PDF, Postscript oder SGML.

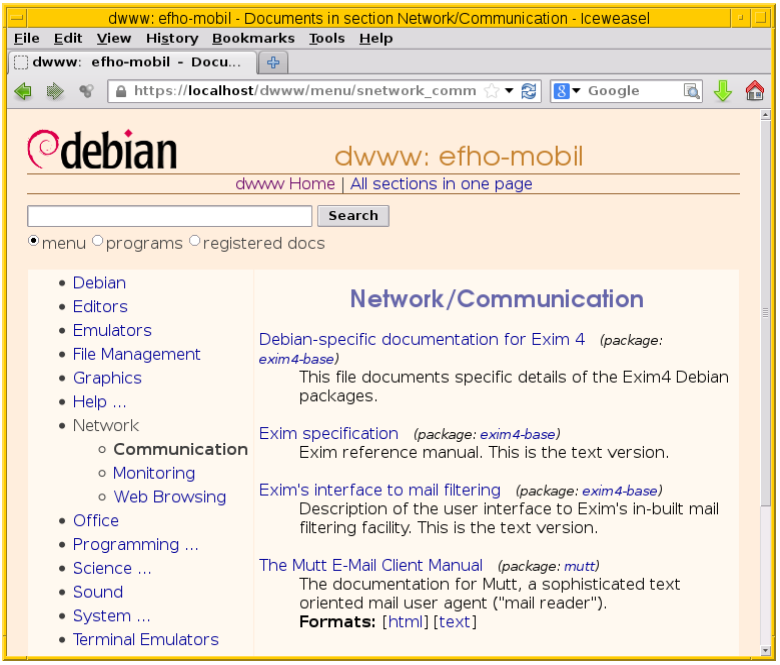

Abbildung 6.15: Auswahl der Hilfeseiten zum Menüpunkt Netzwerkkommunikation

Wählen Sie ein Paket aus der Liste aus, erhalten Sie detailliertere Informationen dazu. Dieses zeigt Abbildung 6.16 anhand des Programms Synaptic.

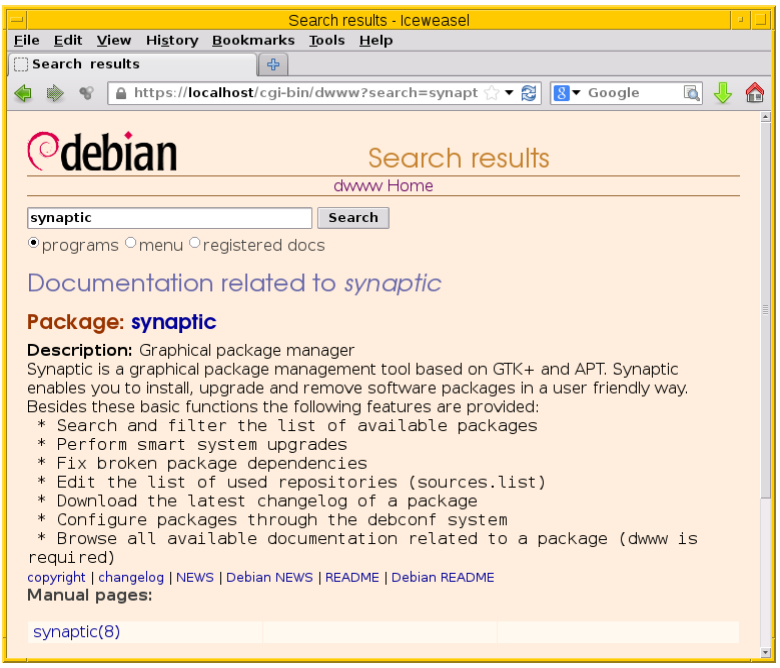

Abbildung 6.16: Auswahl der Hilfeseiten zu Synaptic

`dpkg-www` verfügt auch über eine sekundäre Schnittstelle — die **Kommandozeile**. Darüber fragen Sie sowohl Informationen zu ihrem eigenen System, als auch zu entfernten Rechnern ab. Letzteres gelingt nur, sofern dort `dpkg-www` auch installiert und über das Netzwerk erreichbar ist. Nutzen Sie den Apache Webserver, muss diese Funktionalität zuvor in der Datei `/etc/apache2/conf.d/dpkg-www` freigeschaltet worden sein.

Neben dem Paketnamen versteht das Programm die folgenden beiden Optionen:

`-s` (Langform `--stdout`)
: die Ausgabe erfolgt nicht im Webbrowser, sondern auf dem Terminal.

`-h Hostname`
: Hostname des angefragten Rechners.

Für das Paket *bash* auf dem *lokalen Rechner* und der späteren Ausgabe im Webbrowser nutzen Sie den nachfolgenden Aufruf. Dazu aggregiert `dpkg-www` nacheinander die Ergebnisse drei Kommandos `dpkg-query -S bash`, `dpkg-query -l bash` und `dpkg-query -L bash` zur Paketsuche und zur Bestimmung der Dateien in dem angefragten Paket (siehe auch Abschnitt 8.23). Zur Ausgabe im Webbrowser wertet `dpkg-www` die Umgebungsvariable `$BROWSER` aus, startet das darüber benannte Programm und öffnet ein zusätzliches Fenster zur Darstellung (siehe dazu auch Kapitel 18).

Lokale Hilfedokumente zum Paket bash mittels dpkg-www anzeigen

```
$ dpkg-www bash
$
```

Um die Informationen zu dem gleichen Paket zuerhalten, welches jedoch auf dem entfernten Rechner mit dem Hostnamen `kiste` installiert ist, funktioniert dieser Aufruf mit der zusätzlichen Option `-h`:

Entfernte Hilfedokumente zum Paket bash mittels dpkg-www im Webbrowser anzeigen

```
$ dpkg-www -h kiste bash
$
```

Möchten Sie diese Informationen stattdessen auf ihrem aktuellen Terminal ausgeben, ergänzen Sie den obigen Aufruf um den Parameter `-s` wie folgt (hier am Beispiel des Pakets *htop*):

Entfernte Hilfedokumente zum Paket htop mittels dpkg-www im Terminal anzeigen

```
$ dpkg-www -h kiste -s htop
Package: htop
Status: install ok installed
Priority: optional
Section: utils
Installed-Size: 195
Maintainer: Eugene V. Lyubimkin <jackyf@debian.org> [Debian Bug Report]
Architecture: i386
Version: 1.0.1-1
Depends: libc6 (>= 2.3.4), libncursesw5 (>= 5.6+20070908), libtinfo5
Suggests: strace, ltrace
Description: interactive processes viewer
 Htop is an ncursed-based process viewer similar to top, but it
 allows one to scroll the list vertically and horizontally to see
 all processes and their full command lines.

 Tasks related to processes (killing, renicing) can be done without
 entering their PIDs.

Homepage: http://htop.sourceforge.net

Files owned by package htop:

/usr
/usr/bin
/usr/bin/htop
/usr/share
/usr/share/applications
/usr/share/applications/htop.desktop
/usr/share/doc
/usr/share/doc/htop
/usr/share/doc/htop/AUTHORS
/usr/share/doc/htop/README
/usr/share/doc/htop/changelog.Debian.gz
/usr/share/doc/htop/changelog.gz
/usr/share/doc/htop/copyright
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/htop.1.gz
/usr/share/menu
/usr/share/menu/htop
/usr/share/pixmaps
/usr/share/pixmaps/htop.png
$
```

6.5.2 Ubuntu Landscape

Canonical stellt mit Ubuntu Landscape [Ubuntu-Landscape] seit 2008 eine zentrale Administrationsoberfläche für Einzelrechner und Serversysteme bereit. Ubuntu Landscape ist Teil des kostenpflichtigen Servicepakets Ubuntu Advantage und sowohl als gehosteter Dienst nutzbar oder auch lokal installierbar[10]. Darüberhinaus steht eine API zur individuellen Ansteuerung bereit.

Die erhoffte Nutzergruppe von Ubuntu Landscape sind weniger die Endbenutzer, sondern Unternehmen und deren gesamte Infrastruktur. Das Ziel besteht dabei in der möglichst vollständigen, weitestgehend automatisierten und zentralen Systemadministration für bis zu 40000 Computer. Das Angebot umfasst daher eine webbasierte Schnittstelle für das Patch- und Änderungsmanagement, die rollenbasierte Verwaltung für Einzelbenutzer und Gruppen sowie Sicherheitseinstellungen von Ubuntu-Desktop-Systemen und -Servern (siehe Abbildung 6.17).

[10] Die Pakete dafür heißen *landscape-client*, *landscape-client-ui*, *landscape-client-ui-install* und *landscape-common*

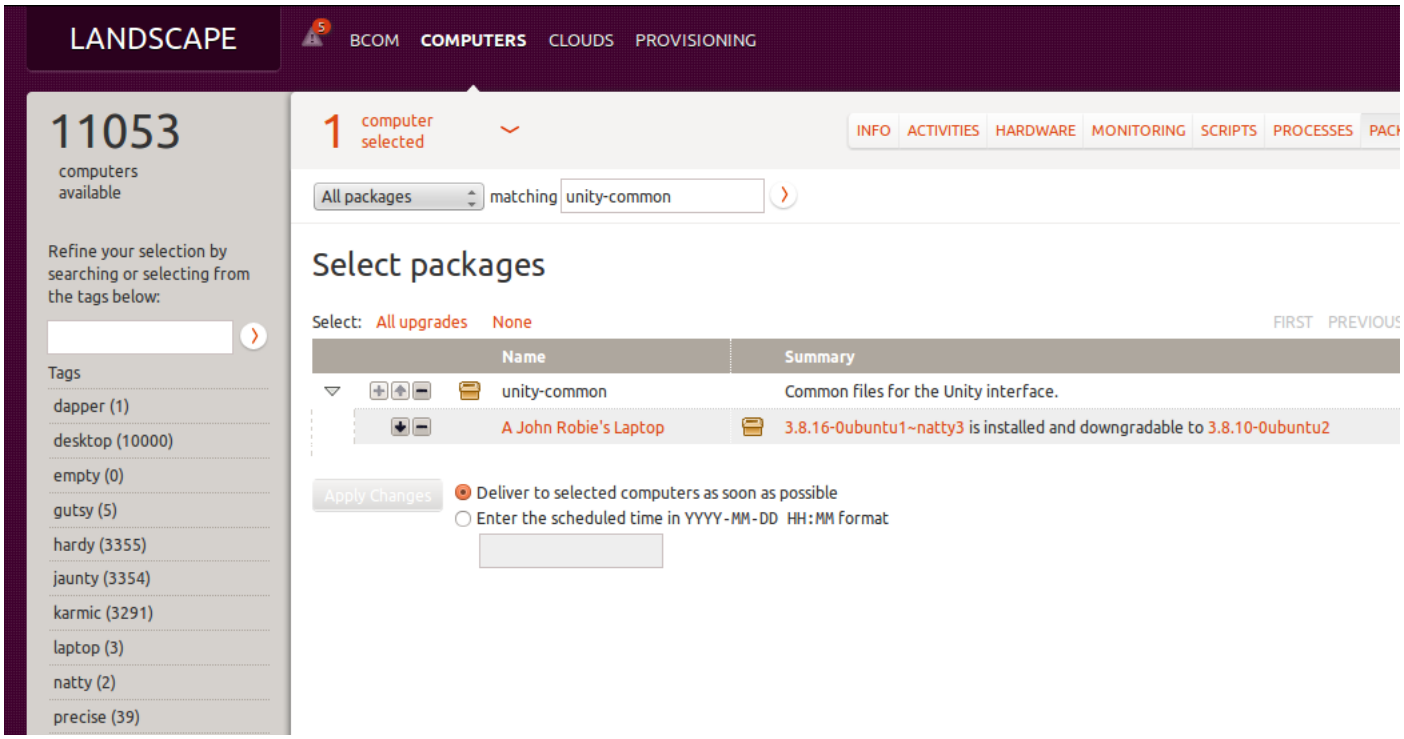

Abbildung 6.17: Rollback eines ausgewählten Pakets

Im Bereich der Paketverwaltung und Systemaktualisierung erhalten Sie einerseits Informationen zu den einzelnen Softwarepaketen und deren Verfügbarkeit und sehen auf den betreuten und über Ubuntu Landscape gepflegten Computern deren Installationsstatus. Wie bei Communtu (siehe Abschnitt 6.5.4) kommen auch hier Metapakete (siehe Abschnitt 2.7.2) verstärkt zum Einsatz. Darüber werden Paketgruppen und ganze Softwareprofile für die betreuten System realisiert und abgebildet.

6.5.3 Appnr

Seit 2008 betreibt der Japaner Akira Ohgaki in Eigenregie seine nichtkommerzielle Plattform Appnr [appnr]. Darüber stehen `deb`-Pakete verschiedener Repositories bereit, die Sie für ihr jeweiliges Ubuntu-Linuxsystem beziehen und installieren können (siehe Abbildung 6.18). Der Name des Projekts orientiert sich am Neusprech „App“, da Sie die Installation im Webbrowser beginnen und die Handhabung der Nutzung einer „App“ auf Geräten mit einem berührungsempfindlichen Bildschirm entspricht.

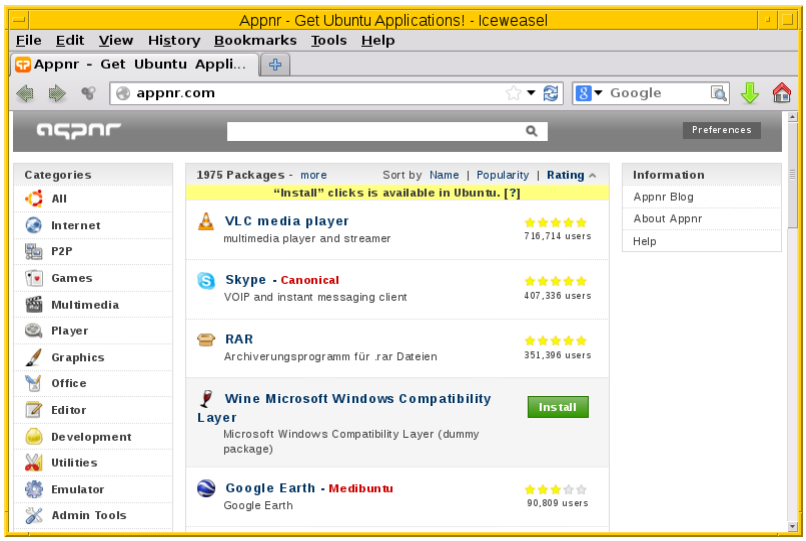

Abbildung 6.18: Vorauswahl für Appnr im Webbrowser *Iceweasel*

Aus technischer Sicht verfügt Appnr nicht über ein eigenes Repository. Stattdessen stellt der angebotene Dienst eine browserbasierte, übersichtliche Schnittstelle für Repositories von Ubuntu und verschiedenen Drittanbietern bereit, die entsprechend dem aktuellen Zeitgeschmack gestaltet ist. In der linken Spalte der Darstellung finden Sie die Paketkategorien und in der mittleren Spalte die dazugehörigen Softwarepakete. Über das Eingabefeld am oberen Rand spezifizieren Sie ihren Suchbegriff, nach welchem danach sowohl in den Paketnamen, der Paketbeschreibung, als auch in der Liste der enthaltenen Dateien gesucht wird.

Im Ergebnis sehen Sie eine Kurzbeschreibung zum Paket, Bildschirmfotos von `screenshots.debian.net` [screenshots.debian.net] (sofern hinterlegt und verfügbar), eine Bewertung und eine Information zur Häufigkeit der Installation. Letzteres bezieht sich auf die Informationen, die Ubuntu im Rahmen seiner statistischen Erhebung durch den *Popularity Contest* gesammelt hat.

Zum Bezug der ausgewählten Pakete und sofortiger Installation benötigen Sie das Paket *apturl* ([Ubuntu-apturl] und [Vogt-apturl]), welches wir Ihnen in Kapitel 24 genauer vorstellen.

6.5.4 Communtu

Communtu ist eine Plattform, die seit 2008 vom Bremer Verein *Natur, Geist und Technik — Verein zur Förderung der Allgemeinbildung e.V.* betrieben und gepflegt wird. Über die Webseite stellen Sie sich aufgabenbezogene `deb`-Pakete für Ubuntu zusammen, die Sie danach auf ihrem Linuxsystem einspielen können. Dabei löst die Paketverwaltung alle Abhängigkeiten zwischen den einzelnen Paketen auf. Den Quellcode der Plattform hat der Verein unter der Affero GNU Public License (AGPL) freigegeben. Für Debian GNU/Linux gibt es seit 2004 ein ähnliches Konzept unter dem Namen *Debian Pure Blends* (siehe [Debian-Pure-Blends]).

Hintergrund ist die Idee, alle benötigten Programme für eine Aufgabe zu kombinieren und als Paketbündel abzulegen. Ein solches „Bündel“ ist stets einer Kategorie zugeordnet, beispielsweise Audio, Büro oder Komplettinstallation (siehe Abbildung 6.19). Die Webseite stellt die Infrastruktur zur Zusammenstellung bereit und ermöglicht Ihnen einerseits, Detailinformationen zu den Paketbündeln abzurufen, erstellte Paketbündel zu hinterlegen und damit auch anderen Benutzer zugänglich zu machen sowie bestehende Paketbündel zu beziehen, online zu bewerten und auch zu verbessern.

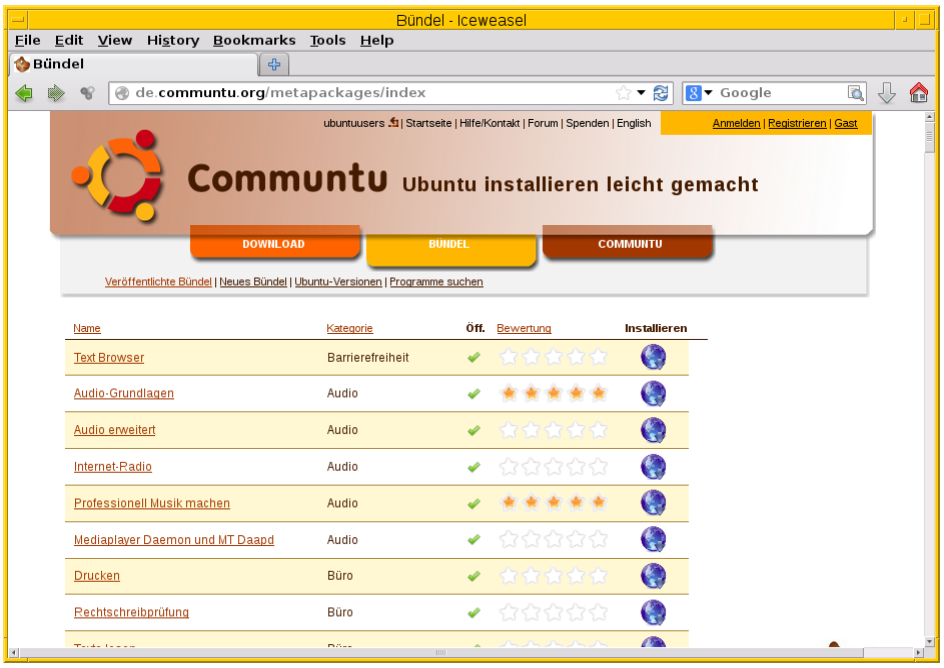

Abbildung 6.19: Verfügbare Bündel auf der Webseite

Es „vereinfacht [Ihnen] damit die Auswahl und Installation von Software, die nicht durch die Standardinstallation des Ubuntu-Systems abgedeckt ist“ (Quelle: UbuntuUsers-Wiki [Communtu]). Zudem integriert es auch Pakete von Fremdquellen und erleichtert somit die sich wiederholende Installation gleichartiger Systeme, bspw. für einen Klassenraum in einer Schule.

Die technische Basis hinter dem Angebot der Plattform bilden Metapakete (siehe Abschnitt 2.7.2) sowie Kapitel 24). Nachdem Sie über die Webseite Ihre gewünschte Software ausgewählt haben, wird automatisch ein erstes passendes Metapaket erzeugt. Dieses beinhaltet lediglich die Abhängigkeiten zu den von Ihnen zuvor ausgewählten Softwarepaketen. Gleichzeitig stellt die Plattform ein weiteres, spezifisches Metapaket bereit, welches ihre Liste der Paketquellen entsprechend anpasst, um damit die Anbindung von weiteren Paketquellen zu ermöglichen. Mittels *apturl* beziehen Sie die erzeugten Metapakete über ihrem Webbrowser, so dass die Paketverwaltung diese Pakete einspielen kann.

6.5.5 Univention Corporate Server (UCS)

Der Univention Corporate Server (UCS) [UCS] ist ein auf kommerzieller Basis vertriebenes OpenSource-Produkt für Infrastrukturlösungen in Unternehmen und Behörden. Neben dem Univention (Linux) Client werden alle „domänenfähigen Betriebssysteme“ (Windows, Mac OS, Linux) via Samba 4 unterstützt.

Die Erweiterung *UCS at School* ist für den Einsatz in Lehr- und Forschungseinrichtungen verfügbar, welche spezielle Softwarekomponenten für die Unterstützung des Lehrbetriebes enthält und damit die Infrastruktur für Bildungseinrichtungen abdeckt. Dazu zählt bspw. Klassenraummanagement, Verteilung und Einsammeln von Arbeitsmaterialien sowie die Moderation von Druckern und Druckaufträgen. Die Entwicklung von UCS begann 2002, der Server ist seit 2004 und der Client seit 2009 verfügbar.

Basis des vom Bremer Unternehmen Univention GmbH zusammengestellten, entwickelten und vertriebenen Produktes ist Debian GNU/Linux, welche um einige Komponenten erweitert wurde. Mit diesen Komponenten werden Skalierbarkeit, Konfigurationsmanagement und weitere Möglichkeiten zur Verfügung gestellt, die in dieser Form in der Linux-Distribution nicht vorhanden sind. Durch regionale Kooperationspartner vor Ort und den Hersteller Univention GmbH werden die Kunden technisch betreut und der Support sichergestellt.

Den Ausgangspunkt bildet stets ein aktuelles Debian — derzeit die Version 7 *Wheezy*. Für diese Softwarebasis pflegt Univention eine Reihe von Eigenentwicklungen und Anpassungen.

Konkret zeigt sich das bereits im Paketnamen. Alle von Univention modifizierten Pakete tragen das Präfix *univention*, so bspw. *univention-dns* zur Einrichtung des Domain Name Systems. Weiterhin besteht eine geänderte Konfiguration der bereitgestellten

Softwarepakete über zusätzliche Vorlagen, sogenannte *univention templates*. Diese Vorlagen und deren Einstellungen richten sich nach der von Ihnen vergebenen Systemrolle und damit der Funktion der jeweiligen, spezifischen UCS-Instanz.

Die **Paketverwaltung** ist mehrstufig und setzt auf den bereits bewährten und beschriebenen Mechanismen von Debian mittels `dpkg` und APT auf. Univention ergänzt diese Werkzeuge um eine zusätzliche Ebene, um die Softwarepakete passend zur vorher festgelegten Systemrolle und den Vorlagen zu installieren und automatisch konfigurieren zu können.

Die Basis bildet der *Univention Installer*, welcher im Prinzip ein modifizierter Debian-Installer ist. Bei der Grundinstallation wählen Sie danach ihre gewünschten Softwarekomponenten über ein modifiziertes Tasksel Abschnitt 6.3.1 aus. Installieren Sie zu einem späteren Zeitpunkt Software nach, stehen Ihnen auf der **Kommandozeile** die beiden Werkzeuge `univention-install` und `univention-remove` zur Verfügung. Diese akzeptieren Paketnamen als Parameter und versorgen `dpkg` und APT mit den zusätzlichen Daten aus den dazugehörigen, von Univention bereitgestellten Vorlagen.

`dpkg` und APT können Sie jederzeit für die grundlegenden Paketoperationen nutzen, um beispielsweise nach Paketen zu suchen, deren Installationszustand zu erfragen oder um die Paketinhalte anzuzeigen. Installieren Sie hingegen Pakete mit `dpkg` und APT eigenhändig nach, müssen Sie die Angaben aus den Vorlagen selbständig in die Konfiguration des Pakets und die zentrale Konfigurationsdatei namens *Univention Configuration Registry (UCR)* übertragen.

Erweiterungen und Werkzeuge von Drittanbietern wählen Sie webbasiert über das sogenannte *Univention App Centre* (siehe Abbildung 6.20) aus. In der darüber angebotenen Paketmenge sind Programme enthalten, die zuvor von Univention freigegeben wurden. Das beinhaltet bspw. Software für Wikis, Groupwares oder auch Werkzeuge zur Inventarisierung. Die Konfiguration der Pakete erfolgt nicht direkt über die Webschnittstelle, sondern ist programmspezifisch. Das gilt auch für Aktualisierungen der Pakete, die hier nicht Updates, sondern *Errata* heißen (siehe [univention-errata]).

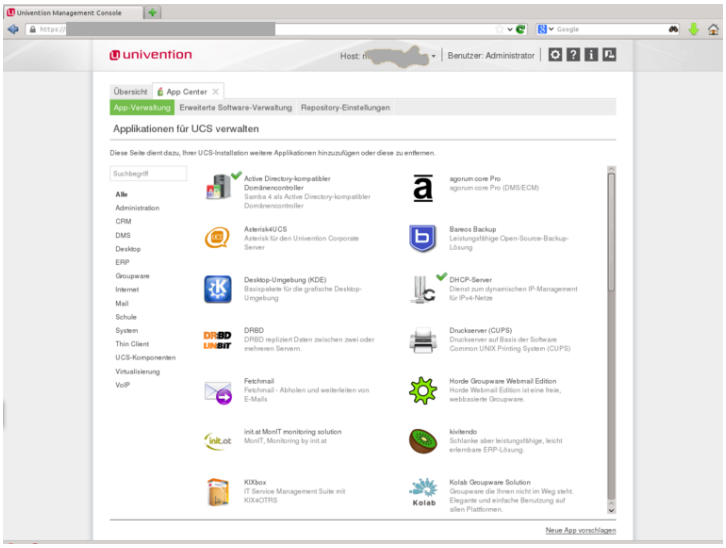

Abbildung 6.20: Univention App Centre

Kapitel 7

Paketcache

7.1 Hintergrundwissen

Die deutsche Übersetzung zum Wort *cache* ist Zwischenspeicher oder Puffer. In der Manpage von `apt-get` wird dafür auch der Begriff *lokales Depot* verwendet.

Laden Sie mittels der APT-Infrastruktur Debian-Pakete vom Spiegelserver herunter, werden diese nicht sofort entpackt, sondern zunächst lokal zwischengespeichert („gecacht“). Vollständig heruntergeladene Pakete liegen im Verzeichnis `/var/cache/apt/archives/`, hingegen nur teilweise heruntergeladene Pakete unter `/var/cache/apt/archives/partial/`.

Erst wenn alle zur Installation oder Aktualisierung benötigten Pakete von APT oder `aptitude` heruntergeladen wurden und auch im Paketcache liegen, wird mit dem Auspacken und Installieren der Pakete begonnen. So ist garantiert, dass alle durch Abhängigkeiten notwendigen Pakete auch lokal verfügbar sind und nichts mehr fehlt. Die Pakete werden daraufhin von `dpkg` unter Verwendung der Bibliotheken (siehe Kapitel 5) ausgepackt, die Dateien an die angegebene Stelle im Verzeichnisbaum kopiert und abschließend ggf. noch automatisch konfiguriert (siehe dazu Abschnitt 8.38).

7.1.1 Was passiert, wenn nicht alle Pakete heruntergeladen werden konnten?

Es kann jedoch vorkommen, dass das Herunterladen eines oder mehrerer Pakete fehlschlägt. Ursachen können beispielsweise sein, dass die Netzwerkverbindung unterbrochen oder der Spiegelserver neugestartet wurde. Möglich ist auch, dass just zwischen dem letzten Aufruf von `apt-get update` und dem Herunterladen der Pakete eine Aktualisierung des Paketspiegels stattfindet und genau das Paket durch ein neueres ersetzt wird, welches Sie gerade zum installieren oder aktualisieren herunterladen möchten.

`apt-get` bricht in diesem Fall ab (TODO: Verifizieren), `aptitude` fragt Sie hingegen als Benutzer, ob Sie trotzdem fortsetzen oder abbrechen möchten. Zu überlegen ist das beispielsweise, wenn nur ein einziges Paket fehlschlug, welches von den anderen unabhängig ist.

Wenn die Netzwerkverbindung (wieder) in Ordnung ist, beheben Sie eine solche Situation in den meisten Situationen ohne viel Aufwand. Das gilt inbesondere aber im letztgenannten Fall. Mit einem weiteren Aufruf von `apt-get update` bringen Sie die Paketlisten auf aktuellen Stand und starten die geplante Aktualisierung oder Installation von Paketen danach nochmals.

7.2 Paketcache-Status

Den aktuellen Zustand des Paketcaches erfahren Sie mit Hilfe das Kommandos `apt-cache stats`. Es wertet den Paketcache hinsichtlich der gefundenen Symbole aus — d.h. die Namen, die Varianten und die Bezüge zwischen den Paketen, die sich derzeit im Cache befinden.

Informationen zum Zustand des Paketcache ausgeben

```
$ apt-cache stats
Gesamtzahl an Paketnamen: 47488 (950 k)
Gesamtzahl an Paketstrukturen: 47488 (2.279 k)
  davon gewöhnliche Pakete: 35987
  davon rein virtuelle Pakete: 371
  davon einzelne virtuelle Pakete: 4324
  davon gemischte virtuelle Pakete: 1029
  davon fehlend: 5777
Gesamtzahl an unterschiedlichen Versionen: 37547 (2.403 k)
Gesamtzahl an unterschiedlichen Beschreibungen: 87385 (2.097 k)
Gesamtzahl an Abhängigkeiten: 222388 (6.227 k)
Gesamtzahl an Version/Datei-Beziehungen: 40866 (654 k)
Gesamtzahl an Beschreibung/Datei-Beziehungen: 87385 (1.398 k)
Gesamtzahl an Bereitstellungen: 7563 (151 k)
Gesamtzahl an Mustern: 164 (1.732 )
Gesamtmenge des Abhängigkeits-/Versionsspeichers: 911 k
Gesamtmenge an Slack: 73,0 k
Gesamtmenge an Speicher: 11,8 M
$
```

Ist der Platz auf ihrem Speichermedium knapp, sehen Sie in der letzten Zeile der Ausgabe, welche Menge durch den Paketcache gerade belegt wird. Wie Sie darin wieder für Ordnung sorgen, lesen Sie unter „Paketcache aufräumen“ in Abschnitt 7.3 nach.

7.3 Paketcache aufräumen

7.3.1 Weshalb aufräumen?

Der Paketcache belegt Speicherplatz auf dem Datenträger ihres Debian-Systems. Da dieser Bereich lediglich temporäre Daten beinhaltet, raten wir Ihnen, diesen regelmäßig aufzuräumen.

Sie finden den Paketcache im Verzeichnis `/var`. Je nach Partitionierung ihres Datenträgers ist der Bereich nicht separat und somit Bestandteil der `/`-Partition. Wenn diese vollständig mit Daten gefüllt ist, funktioniert vieles auf ihrem Linuxsystem nicht mehr.

Bei ausgefeilteren Installationen bestehen häufig separate Partitionen für `/var`, `/var/cache` oder gar `/var/cache/apt/archives`. Läuft nur einer dieser Bereiche voll, können im einfachsten Fall keine weiteren Pakete zwischengespeichert und auch nicht temporär entpackt werden. Falls der Bereich `/var` vollständig belegt ist, können auch keine Logdateien mehr angelegt oder weitere Informationen daran angehängt werden.

Einen Ausweg aus diesem Dilemma gelingt Ihnen mit dem Anlegen einer separaten Partition für `/var/cache/apt/archives`. Wie Sie diese einrichten und mit welchen Vor- und Nachteilen dieser Schritt verbunden ist, zeigen wir Ihnen unter „Cache-Verzeichnis auf separater Partition“ in Kapitel 27.

7.3.2 Kommandos zum Aufräumen

Die Programme `dpkg`, `apt-get`, `aptitude` und Synaptic räumen den Paketcache in der Standardeinstellung nicht eigenständig auf. Ob überhaupt, wann und insbesondere wie aufgeräumt wird, entscheiden Sie als Systembetreuer selbst und müssen dazu das Werkzeug entsprechend konfigurieren.

Grundsätzlich gibt es mehrere, unterschiedlich radikale Ansätze zum Aufräumen des Paketcaches. Die **Variante eins** umfasst das Löschen von Paketen aus dem Cache, die in keinem der verwendeten APT-Repositories mehr verfügbar sind. Die beiden Aufrufe `apt-get autoclean` und `aptitude autoclean` implementieren diesen Ansatz. Bei Synaptic nutzen Sie dazu den Menüpunkt Einstellungen → Dateien und selektieren im Dialogfenster den Eintrag „Nur Pakete löschen, die nicht länger verfügbar sind“ (siehe Abbildung 7.1). In der Standardeinstellung erleichtern diese Kommandos den Cache auch um Pakete, die Sie in der vorliegenden Version nicht mehr vom Spiegelserver beziehen können, die aber noch auf ihrem Linuxsystem installiert sind.

Um hingegen lediglich die Pakete aus dem Cache zu löschen, die auch nicht, oder nicht mehr installiert sind, hilft Ihnen die Konfigurationsoption `APT::Clean-Installed=off` in der Konfiguration von APT. Alternativ teilen Sie das `apt-get` mit diesem Aufruf explizit mit:

Aufruf von apt-get mit ausdrücklicher Konfiguration

```
# apt-get -o APT::Clean-Installed=off autoclean
...
#
```

Allerdings besteht keine Möglichkeit mehr, diese Pakete mittels APT zu einem späteren Zeitpunkt wieder zu installieren, da sie ja in keiner Paketliste mehr vorkommen. Benötigen Sie eines der Pakete später doch wieder, hilft Ihnen nur der Rückgriff auf `dpkg` in Kombination mit dem dazugehörigen `deb`-Paket weiter. Dazu benutzen Sie den Aufruf `dpkg -i Paketname`. `Paketname` bezeichnet den Namen und Pfad der lokalen Datei für das benötigte `deb`-Paket (siehe Abschnitt 8.36).

Variante zwei umfasst das Löschen sämtlicher Pakete aus dem Paketcache — damit schaffen Sie radikal Platz. Die passenden Kommandos dazu lauten `apt-get clean` und `aptitude clean`. Übrig bleiben danach nur die beiden Verzeichniseinträge für die Sperrdatei `lock` und das Unterverzeichnis `partial`, in dem die Fragmente für Pakete landen, die nur teilweise bezogen wurden.

Aufräumen mittels apt-get clean

```
# apt-get clean
# ls /var/cache/apt/archives/
lock  partial
#
```

Bei Synaptic bildet zunächst der Eintrag Einstellungen → Dateien den Ausgangspunkt. Über den Knopf „Alle Paketdateien im Zwischenspeicher löschen“ lösen Sie die Aufräumaktion aus (siehe Abbildung 7.1).

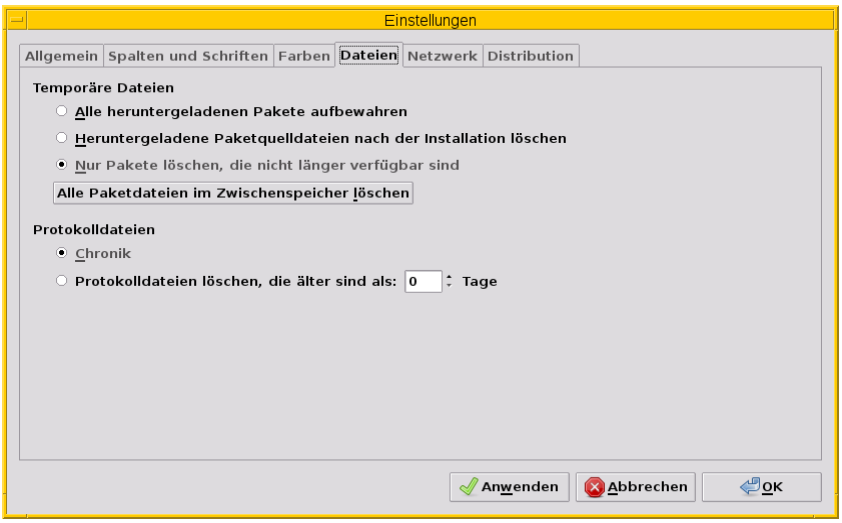

Abbildung 7.1: Großreinemachen mit Hilfe von Synaptic

Selbstverständlich können Sie auch als **Administrator** agieren und dabei gezielt nur ausgewählte oder auch alle vorliegenden `deb`-Dateien manuell aus dem Verzeichnis `/var/cache/apt/archives/` mittels `rm Paketdatei` löschen. Gerade bei den Paketen, die Daten für Spiele beinhalten — z.B. *0ad-data* mit ca. 530 MB Paketdateigröße —, reicht es oft aus, diese einzelnen Dateien aus dem Paketcache zu entfernen, um dort wieder ausreichend Platz zu haben.

Alle derzeit von Debian unterstützten Versionen von APT klagen nicht, wenn Sie das gesamte Verzeichnis `/var/cache/apt/archives/partial/` klammheimlich hinter dem Rücken der beiden Programme einfach komplett entsorgen. APT und `aptitude` legen es bei einem späteren Bedarf einfach von selbst wieder an. Anders sieht es hingegen bei älteren Veröffentlichungen wie z.B. Debian 4.0 *Etch* oder Debian 5.0 *Lenny*, Ubuntu 10.04 LTS *Lucid Lynx* sowie Debian-Derivaten aus der Zeit vor

Mitte 2010 aus. Beachten Sie bitte, dass APT vor Version 0.8 beim Löschen eines der beiden Verzeichnisse `/var/cache/apt/archives/partial/` oder `/var/lib/apt/lists/partial/` dann einfach den Dienst verweigert. Sie beheben das Problem flink, indem Sie die genannten Verzeichnisse manuell wieder anlegen. Haben Sie `/var/cache/` als `tmpfs`-Dateisystem eingehängt (siehe Kapitel 27), so können Sie mit dem Aufruf `mkdir -p /var/cache/apt/archives/partial` als Eintrag in der Datei `/etc/rc.local` dauerhaft Abhilfe schaffen.

7.3.3 Empfehlungen zum Zeitpunkt des Aufräumens

Wann Sie am besten aufräumen, hängt etwas von der Nutzung und dem verfügbaren Plattenplatz ab. In den meisten Fällen ist *nach* dem Installieren und Aktualisieren der Pakete ein guter Zeitpunkt. `aptitude` bietet dies sogar über die Option `Aptitude::Autoclean-After-Update` an (siehe unten).

Ist jedoch der Plattenplatz recht knapp, so kann auch es auch helfen, den Cache bereits *vor* dem Installieren und Aktualisieren aufzuräumen. Das ist insbesondere dann sinnvoll, wenn Sie dies selbst nicht regelmäßig machen und diese Aktion stattdessen per Cron-Job oder über die Konfiguration der Paketverwaltung ausführen lassen. Es macht jedoch keinen Sinn, wenn Sie beispielsweise gleichzeitig die APT-Option `APT::Periodic::Download-Upgradeable-Packages` eingeschaltet haben und damit nachts automatisch alle aktualisierbaren Pakete herunterladen lassen. Leeren Sie den Paketcache danach mit `apt-get clean` komplett, hat das zur Folge, dass die frisch bezogenen Pakete wieder gelöscht werden und ein nachfolgendes `apt-get upgrade` diese erneut herunterladen muss.

7.3.4 Automatisch und regelmäßig Aufräumen

Das manuelle Aufrufen der o.g. Kommandos kostet Zeit. Daher bieten APT und `aptitude` unterschiedliche Möglichkeiten, um diese Vorgänge zu automatisieren.

Das Paket *apt* bringt mit dem Skript `/etc/cron.daily/apt` einen Cron-Job mit, der diverse Aufgaben einmal pro Tag ausführen kann. Konfiguriert wird das Skript ebenfalls über die Datei `/etc/apt/apt.conf`. Den Paketcache betreffen die beiden Einstellungen `APT::Periodic::Download-Upgradeable-Packages` und `APT::Periodic::AutocleanInterval`.

Einstellung `APT::Periodic::Download-Upgradeable-Packages`
: Damit legen Sie die Regelmäßigkeit der Aktualisierung fest. Setzen Sie den Wert auf `1`, so füllt der Cron-Job den Paketcache einmal pro Tag, falls Paketaktualisierungen verfügbar sind. Setzen Sie den Wert hingegen auf `7`, so lädt er verfügbare Paketaktualisierungen nur einmal die Woche herunter. Der Wert `0` (Null) ist die Standardeinstellung und deaktiviert die Funktionalität vollständig.

Einstellung `APT::Periodic::AutocleanInterval`
: Damit regeln Sie die Häufigkeit, mit der das Kommando `apt-get autoclean` ausgeführt wird. Auch hier steht der Wert für den Abstand in Tagen zwischen zwei Ausführungen. Der Wert `0` (Null) schaltet das nächtliche Aufräumen ganz ab und ist auch die Standardvorgabe.

Die Dokumentation zu diesem Skript finden Sie in den Kommentarzeilen am Anfang der Datei `/etc/cron.daily/apt`. Dort finden sich noch weitere und feinere Einstellmöglichkeiten zum automatischen Aufräumen des Paketcaches, z.B. anhand des Alters der Pakete.

`aptitude` dagegen bietet eine Zeitsteuerung über Schalter und Optionen an. Damit erfolgt das Aufräumen via `autoclean` oder `clean` vor oder nach der Installation von Paketen automatisch:

Schalter `--clean-on-startup`
: entspricht dem Aufruf `aptitude autoclean`

Schalter `--autoclean-on-startup`
: entspricht dem Aufruf `aptitude clean`

Ähnliches ermöglicht Ihnen `aptitude` auch über die Text-Modus-Bedienoberfläche. Setzen Sie in den Einstellungen unter „Veraltete Paketdateien nach dem Laden von neuen Paketlisten löschen“ ein Häkchen, entspricht das der Konfigurationsoption `Aptitude::AutoClean-After-Update`. Damit löscht Aptitude nach jeder Aktualisierung der Paketlisten (durch `aptitude`) alle Paketdateien aus dem Paketcache, die nicht mehr von einem in `/etc/apt/sources.list` aufgeführten Paketmirror heruntergeladen werden können.

Kapitel 8

Paketoperationen

8.1 Paketoperationen und deren Abfolge

Als Paketoperation verstehen wir hier im Buch ein Kommando oder einen Aufruf eines Programms zur Paketverwaltung, mit dem Sie entweder den aktuellen Paketbestand ausgeben, in diesem Paketbestand nach bestimmten Kriterien suchen oder Änderungen im Paketbestand veranlassen. Für letzteres heißt das bspw. Paketlisten aktualisieren oder modifizieren, Pakete hinzufügen und auch Pakete wieder deinstallieren.

Wir besprechen die Paketoperationen in gleicher Reihenfolge. Den Anfang machen Aufrufe ohne Änderung des Paketbestands, d.h. bspw. Statusinformationen erhalten und die Recherche nach bestimmten Kriterien (Abschnitt 8.3 bis Abschnitt 8.31). Daran schließen sich Aufrufe an, die den Paketbestand verändern, d.h. neue Pakete hinzufügen und einrichten sowie Pakete deinstallieren und der Umgang mit Waisen (Abschnitt 8.32 bis Abschnitt 8.44). Den Abschluss bildet der Vorgang, eine ganze Distribution auf den neuen Stand zu bringen oder zu einer anderen Veröffentlichung zu wechseln (siehe Abschnitt 8.45).

8.2 Paketlisten und Muster

Auch wenn bei den nachfolgend vorgestellten Paketoperationen und Beispielen stets `Paketname` im Singular steht, können Sie beim Aufruf einen oder mehrere exakte Paketnamen angeben. Diese Liste trennen Sie durch Leerzeichen, wobei die Reihenfolge der genannten Pakete im Allgemeinen keine Rolle spielt. Nachfolgend soll das an einem Aufruf mit `apt-get` zur Installation der Pakete *xpdf*, *kile* und *cssed* deutlich werden.

Installation der Pakete xpdf, kile und cssed mittels APT in einem Aufruf

```
# apt-get install xpdf kile cssed
...
#
```

Zulässig sind ebenfalls ein Fragment eines Namens oder auch ein Paketmuster über einen regulären Ausdruck. Dann werden alle Pakete ausgewählt, die auf das Fragment bzw. Muster passen. Das betrifft insbesondere die Kommandozeilenwerkzeuge `dpkg`, APT, `aptitude` und `ara`. Bei den graphischen Programmen wird das hingegen sehr unterschiedlich gehandhabt.

Auflistung aller Dokumentationspakete zu aptitude über ein Muster

```
# dpkg -l aptitude-doc*
Gewünscht=Unbekannt/Installieren/R=Entfernen/P=Vollständig Löschen/Halten
| Status=Nicht/Installiert/Config/U=Entpackt/halb konFiguriert/
         Halb installiert/Trigger erWartet/Trigger anhängig
|/ Fehler?=(kein)/R=Neuinstallation notwendig (Status, Fehler: GROSS=schlecht)
||/ Name               Version        Architektur    Beschreibung
+++-==================-==============-==============-================================================ ↩

un  aptitude-doc       <keine>                       (keine Beschreibung vorhanden)
```

```
ii  aptitude-doc-cs    0.6.8.2-1      all            Czech manual for aptitude, a terminal- ↩
    base
ii  aptitude-doc-en    0.6.8.2-1      all            English manual for aptitude, a  ↩
    terminal-ba
ii  aptitude-doc-es    0.6.8.2-1      all            Spanish manual for aptitude, a  ↩
    terminal-ba
ii  aptitude-doc-fi    0.6.8.2-1      all            Finnish manual for aptitude, a  ↩
    terminal-ba
ii  aptitude-doc-fr    0.6.8.2-1      all            French manual for aptitude, a terminal ↩
    -bas
ii  aptitude-doc-it    0.6.8.2-1      all            Italian manual for aptitude, a  ↩
    terminal-ba
ii  aptitude-doc-ja    0.6.8.2-1      all            Japanese manual for aptitude, a  ↩
    terminal-b
```

Verwenden Sie das *multiarch*-Feature (siehe Abschnitt 1.2.3), geben Sie hinter dem Paketnamen noch einen Doppelpunkt und danach die gewünschte Architektur an. Damit werden nur die Pakete berücksichtigt, die für die angegebene Architektur bereitstehen. Benennen Sie keine Architektur explizit, werden die Pakete ihrer Systemarchitektur benutzt.

Suche nach Paketen für die Architektur i386

```
# dpkg -l "*:i386"
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version      Architecture Description
+++-==============-============-============-=================================
ii  libc6:i386     2.19-11      i386         GNU C Library: Shared libraries
#
```

8.3 Bekannte Paketnamen auflisten

APT kann Ihnen ausgeben, welche Pakete es überhaupt kennt. Dazu verfügt das Werkzeug `apt-cache` über das Unterkommando `pkgnames`. Die Ausgabe, die Sie nach dessen Aufruf erhalten, hängt von den von Ihnen genutzten Paketquellen und den darüber verfügbaren Paketen ab.

Die nachfolgende Ausgabe listet die Pakete zusätzlich alphabetisch aufsteigend sortiert und seitenweise auf. Im Aufruf kommen dazu die beiden UNIX-Kommandos `sort` und `more` zum Einsatz.

Seitenweise und alphabetisch sortierte Ausgabe der Paketnamen, die APT kennt

```
$ apt-cache pkgnames | sort | more
0ad
0ad-data
0ad-dbg
2ping
2vcard
3270-common
389-console
3dchess
...
$
```

In eine ähnliche Richtung gehen die beiden Werkzeuge `grep-available` und `grep-aptavail` aus dem Paket *dctrl-tools* [Debian-Paket-dctrl-tools]. Beide liefern Ihnen Informationen darüber, ob und in welcher Version das von Ihnen angefragte Paket aus den Paketquellen zur Verfügung steht. Während `grep-available` weitestgehend die Informationen ausgibt, die Sie mittels `dpkg -s` erhalten, liefert Ihnen `grep-aptavail` die vollständigen Informationen, so bspw. auch, wo Sie das Paket in der Verzeichnishierarchie der Paketquellen finden. Nachfolgendes Beispiel zeigt die Recherche anhand des Pakets *xpdf*.

Paketinformationen zu xpdf

```
$ grep-aptavail -PX xpdf
Package: xpdf
Version: 3.03-10
Installed-Size: 447
Maintainer: Michael Gilbert <mgilbert@debian.org>
Architecture: i386
Provides: pdf-viewer
Depends: lesstif2 (>= 1:0.94.4), libc6 (>= 2.4), libgcc1 (>= 1:4.1.1), libpoppler19 (>=  ↩
    0.18.4), libstdc++6 (>= 4.1.1), libx11-6, libxt6
Recommends: poppler-utils, poppler-data, gsfonts-x11, cups-bsd
Description: Portable Document Format (PDF) reader
Homepage: http://www.foolabs.com/xpdf/
Description-md5: fa7a14f325304cc49bbc0086a88d330e
Tag: implemented-in::c++, interface::x11, role::program, scope::application,
 uitoolkit::motif, use::viewing, works-with-format::pdf,
 works-with::text, x11::application
Section: text
Priority: optional
Filename: pool/main/x/xpdf/xpdf_3.03-10_i386.deb
Size: 196690
MD5sum: 02e96c8b0c4b82a496deb2c386ed1042
SHA1: 233ddb2074f1694e8fda75c63fb714464d42e15b
SHA256: 2cced2a99c1b028bd5b2c5e37c3a6206c643167b1a5465481e5621f2d3f390a3
$
```

Kennen Sie jedoch nur ein Fragment eines Paketnamens, gibt es einen üblichen Weg und eine Abkürzung. Der *übliche Weg* kombiniert `apt-cache` und das UNIX-Kommando `grep` über eine Pipe miteinander. *Muster* bezeichnet hier das Ihnen bereits bekannte Fragment. Nachfolgend sehen Sie die Recherche nach dem Muster `aptitude`.

Suche nach dem Muster aptitude

```
$ apt-cache pkgnames | grep aptitude
aptitude
aptitude-common
aptitude-doc-cs
aptitude-doc-fi
aptitude-doc-en
aptitude-doc-es
aptitude-doc-fr
aptitude-doc-ja
aptitude-doc-it
aptitude-dbg
$
```

Die *Abkürzung* führt über das Werkzeug `dglob` aus dem Paket *debian-goodies* [Debian-Paket-debian-goodies]. Es setzt auf `apt-file` auf und liefert eine identische Ausgabe – aber der Tippaufwand ist geringer. Nachfolgende Ausgabe zeigt Ihnen das Suchergebnis zu den Paketen, die das Muster `ttd` beinhalten. Damit finden Sie beispielsweise alle Pakete, die zum Spiel Open Transport Tycoon Deluxe (openttd) gehören.

Die im Aufruf verwendete Option `-a` listet Ihnen alle Pakete in alphabetisch aufsteigender Reihenfolge auf – unabhängig davon, ob diese jeweils auf Ihrem System installiert sind oder nicht. Ohne die Option beschränkt sich `dglob` nur auf die bereits installierten Pakete.

Rechercheergebnis von dglob mit dem Muster ttd

```
$ dglob -a ttd
liblttd-dev
liblttd0
openttd
openttd-data
openttd-dbg
openttd-opengfx
```

```
openttd-openmsx
openttd-opensfx
$
```

8.4 Paketstatus erfragen

Diese Aktion betrifft meist nur ein einzelnes Paket, welches auf dem System installiert ist oder war. Im Alltag ergeben sich mehrere Situationen, in denen das Wissen über den Zustand eines Pakets wichtig ist.

Die Grundfrage ist häufig, ob derzeit ein bestimmtes Paket oder Programm auf Ihrem Linuxsystem installiert ist, und falls ja, in welchem Zustand befindet sich dieses Paket. Es kann vollständig oder nur teilweise installiert sein, liegt in nicht konfiguriertem Status vor oder wurde möglicherweise bereits wieder entfernt. Zu klären ist in dem Fall auch, ob es vollständig entfernt wurde oder noch „Reste“ übrig sind. Dazu zählen die Konfigurationsdateien zu den Paketen bzw. den Programmen daraus. Darüberhinaus ist interessant, welche zusätzlichen Dateien verfügbar sind, bspw. Übersetzungen bzw. Sprachpakete.

Eine kompakte Übersicht erhalten Sie mit Hilfe des Kommandos `dpkg -l` *Paketname*. In Abschnitt 8.5 besprechen wir das genauer. Ausführlicher sind die Ausgaben zu den `dpkg`-Schaltern `-s` und `-I` sowie den Aufrufen von `aptitude show`, `apt-cache show` sowie `apt-mark`.

8.4.1 `dpkg -s` *Paketname* und `dlocate -s` *Paketname*

Mit diesen beiden Aufrufen ermitteln Sie den Status eines installierten Pakets. Die Langform zu `-s` ist `--status`. Intern greift `dpkg` auf `dpkg-query` zurück, so daß Sie die gleichen Schalter verwenden können. Paketname bezeichnet hier den Namen eines Debianpakets.

Die Ausgabe beinhaltet bspw. die Paketfelder Paketname (siehe Abschnitt 2.11), Status, Priorität (siehe Abschnitt 2.13), Paketkategorie (siehe Abschnitt 2.8), installierte Größe, Maintainer, Architektur (siehe Abschnitt 1.2) und Version (siehe Abschnitt 2.11) aus. Darunter listet `dpkg` die dazugehörige, hinterlegte Paketbeschreibung auf.

Der nachfolgende Aufruf ist zudem identisch zu `grep-status -F Package -X htop`, wobei Sie mit `-F` das entsprechende Paketfeld und mit `-X` den Paketnamen angeben. Das Kommando `grep-status` ist Bestandteil des Pakets *dctrl-tools* [Debian-Paket-dctrl-tools].

Status des Pakets htop mittels dpkg ermitteln

```
$ dpkg -s htop
Package: htop
Status: install ok installed
Priority: optional
Section: utils
Installed-Size: 195
Maintainer: Eugene V. Lyubimkin <jackyf@debian.org>
Architecture: i386
Version: 1.0.1-1
Depends: libc6 (>= 2.3.4), libncursesw5 (>= 5.6+20070908), libtinfo5
Suggests: strace, ltrace
Description: interactive processes viewer
 Htop is an ncursed-based process viewer similar to top, but it
 allows one to scroll the list vertically and horizontally to see
 all processes and their full command lines.
 .
 Tasks related to processes (killing, renicing) can be done without
 entering their PIDs.
Homepage: http://htop.sourceforge.net
$
```

8.4.2 `dpkg -I` *deb-Datei*

Im Gegensatz zu `dpkg -s` verarbeitet der Schalter `-I` (Langform `--info`) lokal vorliegende deb-Dateien. Daraus extrahiert `dpkg` bzw. dessen Hilfsprogramm `dpkg-deb` die Einträge der Steuerdatei sowie die Paketinformationen.

Detailinformationen des Pakets htop mittels dpkg ermitteln

```
$ dpkg -I htop_1.0.3-1_amd64.deb
 neues Debian-Paket, Version 2.0.
 Größe 75316 Byte: control-Archiv= 1156 Byte.
     593 Byte,    17 Zeilen      control
     618 Byte,    10 Zeilen      md5sums
     185 Byte,     7 Zeilen   *  postinst             #!/bin/sh
     160 Byte,     5 Zeilen   *  postrm              #!/bin/sh
 Package: htop
 Version: 1.0.3-1
 Architecture: amd64
 Maintainer: Eugene V. Lyubimkin <jackyf@debian.org>
 Installed-Size: 204
 Depends: libc6 (>= 2.15), libncursesw5 (>= 5.6+20070908), libtinfo5
 Suggests: strace, ltrace
 Section: utils
 Priority: optional
 Homepage: http://hisham.hm/htop/
 Description: interactive processes viewer
  Htop is an ncursed-based process viewer similar to top, but it
  allows one to scroll the list vertically and horizontally to see
  all processes and their full command lines.
  .
  Tasks related to processes (killing, renicing) can be done without
  entering their PIDs.
$
```

8.4.3 `apt-cache show` *Paketname*

`apt-cache` ist Bestandteil des Debian-Pakets *apt* [Debian-Paket-apt]. Der Aufruf `apt-cache show` *Paketname* liefert ein ähnliches Ergebnis wie obiges `dpkg -s` *Paketname*, ist jedoch noch ausführlicher. Neben einer übersetzten Paketbeschreibung (Lokalisierung, sofern vorhanden) erscheinen zusätzlich die Debtags (siehe Kapitel 13) zum Paket, der Dateiname und Pfad im Paketmirror und die GPG-Keys zur Validierung des Pakets (siehe Abschnitt 8.35).

Status des Pakets htop mit apt-cache ermitteln

```
$ apt-cache show htop
Package: htop
Version: 1.0.1-1
Installed-Size: 195
Maintainer: Eugene V. Lyubimkin <jackyf@debian.org>
Architecture: i386
Depends: libc6 (>= 2.3.4), libncursesw5 (>= 5.6+20070908), libtinfo5
Suggests: strace, ltrace
Description-de: Interaktiver Prozessbetrachter
 Htop ist ein ncurses-basierter Prozessbetrachter ähnlich wie top, jedoch
 ermöglicht er Ihnen die Liste vertikal und horizontal zu durchlaufen, um
 alle Prozesse und deren vollständige Kommandozeilen zu sehen.
 .
 Mit Prozessen verbundene Aufgaben wie das (zwangsweise) Beenden und die
 Neufestlegung der Priorität können ohne Eingabe der PIDs erledigt werden.
Homepage: http://htop.sourceforge.net
Description-md5: 8eb5aa19b3c92a975dc78e2165f6688d
Tag: admin::monitoring, interface::text-mode, role::program, scope::utility,
 uitoolkit::ncurses, use::monitor, works-with::software:running
```

```
Section: utils
Priority: optional
Filename: pool/main/h/htop/htop_1.0.1-1_i386.deb
Size: 71634
MD5sum: 9a12ed8d648a0b16a08f16aa06a6ee9c
SHA1: 25eb706b210a165efae3a149338c129c383b82df
SHA256: b41970322366d8a8fd174aa32b223dd54d05e4ab1dafddd97390e0fc5f17ed41
$
```

8.4.4 `apt-cache showpkg` *Paketname*

Desweiteren verfügt `apt-cache` über das Unterkommando `showpkg`. Primär dient es dazu, die verfügbaren Paketvarianten samt deren Abhängigkeiten und Übersetzungen darzustellen. Die Ermittlung der einfachen und umgekehrten Paketabhängigkeiten besprechen wir ausführlicher unter „Paketabhängigkeiten anzeigen“ in Abschnitt 8.17.

Die nachfolgenden Ausgaben zeigen die Detailansicht für die Pakete *htop* und *openvpn*. Für ersteres steht nur ein Paket zur Verfügung, bei dem zweiten hingegen eine aktualisierte Variante. Daher umfaßt die Ausgabe zwei Einträge mit den Versionen 2.3.4-5 und 2.3.4-5+deb8u1, wobei die letztgenannte Version noch auf dem Paketmirror liegt.

Detailansicht zum Paket htop via apt-cache showpkg (Debian 7 Wheezy)

```
$ apt-cache showpkg htop
Package: htop
Versions:
1.0.1-1 (/var/lib/apt/lists/ftp.de.debian.org_debian_dists_wheezy_main_binary-i386_Packages ↩
    ) (/var/lib/dpkg/status)
 Description Language:
                 File: /var/lib/apt/lists/ftp.de.debian.org_debian_dists_wheezy_main_binary ↩
                     -i386_Packages
                  MD5: 8eb5aa19b3c92a975dc78e2165f6688d
 Description Language: de
                 File: /var/lib/apt/lists/ftp.de.debian. ↩
                     org_debian_dists_wheezy_main_i18n_Translation-de
                  MD5: 8eb5aa19b3c92a975dc78e2165f6688d
 Description Language: en
                 File: /var/lib/apt/lists/ftp.de.debian. ↩
                     org_debian_dists_wheezy_main_i18n_Translation-en
                  MD5: 8eb5aa19b3c92a975dc78e2165f6688d


Reverse Depends:
  education-common,htop
Dependencies:
1.0.1-1 - libc6 (2 2.3.4) libncursesw5 (2 5.6+20070908) libtinfo5 (0 (null)) strace (0 ( ↩
    null)) ltrace (0 (null))
Provides:
1.0.1-1 -
Reverse Provides:
$
```

Detailansicht zum Paket openvpn via apt-cache showpkg (Debian 8 Jessie)

```
apt-cache showpkg openvpn
Package: openvpn
Versions:
2.3.4-5+deb8u1 (/var/lib/apt/lists/ftp.de.debian.org_debian_dists_jessie_main_binary- ↩
    amd64_Packages)
 Description Language:
                 File: /var/lib/apt/lists/ftp.de.debian.org_debian_dists_jessie_main_binary ↩
                     -amd64_Packages
                  MD5: 2ebe91e411d46309a61861db507e5c2f
```

```
 Description Language: de
                 File: /var/lib/apt/lists/ftp.de.debian. ↩
                     org_debian_dists_jessie_main_i18n_Translation-de
                  MD5: 2ebe91e411d46309a61861db507e5c2f
 Description Language: en
                 File: /var/lib/apt/lists/ftp.de.debian. ↩
                     org_debian_dists_jessie_main_i18n_Translation-en
                  MD5: 2ebe91e411d46309a61861db507e5c2f

2.3.4-5 (/var/lib/dpkg/status)
 Description Language:
                 File: /var/lib/apt/lists/ftp.de.debian.org_debian_dists_jessie_main_binary ↩
                     -amd64_Packages
                  MD5: 2ebe91e411d46309a61861db507e5c2f
 Description Language: de
                 File: /var/lib/apt/lists/ftp.de.debian. ↩
                     org_debian_dists_jessie_main_i18n_Translation-de
                  MD5: 2ebe91e411d46309a61861db507e5c2f
 Description Language: en
                 File: /var/lib/apt/lists/ftp.de.debian. ↩
                     org_debian_dists_jessie_main_i18n_Translation-en
                  MD5: 2ebe91e411d46309a61861db507e5c2f


Reverse Depends:
  openvpn:i386,openvpn
  openvpn-auth-radius,openvpn 2
  openvpn-auth-ldap,openvpn 2
  network-manager-openvpn,openvpn 2.1~rc9
  kvpnc,openvpn
  gadmin-openvpn-server,openvpn
  gadmin-openvpn-client,openvpn
  eurephia,openvpn 2
  collectd-core,openvpn
Dependencies:
2.3.4-5+deb8u1 - debconf (18 0.5) debconf-2.0 (0 (null)) libc6 (2 2.15) liblzo2-2 (0 (null) ↩
    ) libpam0g (2 0.99.7.1) libpkcs11-helper1 (2 1.11) libssl1.0.0 (2 1.0.0) init-system- ↩
    helpers (2 1.18~) initscripts (2 2.88dsf-13.3) iproute2 (0 (null)) openssl (0 (null))  ↩
    resolvconf (0 (null)) easy-rsa (0 (null)) openvpn:i386 (0 (null))
2.3.4-5 - debconf (18 0.5) debconf-2.0 (0 (null)) libc6 (2 2.15) liblzo2-2 (0 (null))  ↩
    libpam0g (2 0.99.7.1) libpkcs11-helper1 (2 1.11) libssl1.0.0 (2 1.0.0) init-system- ↩
    helpers (2 1.18~) initscripts (2 2.88dsf-13.3) iproute2 (0 (null)) openssl (0 (null))  ↩
    resolvconf (0 (null)) easy-rsa (0 (null)) openvpn:i386 (0 (null))
Provides:
2.3.4-5+deb8u1 -
2.3.4-5 -
Reverse Provides:
$
```

8.4.5 `aptitude show` *Paketname*

Das Ergebnis des Aufrufs von `aptitude show` *Paketname* kombiniert die Ausgabe von `dpkg -s` mit Teilen von `apt-cache show`. Hervorzuheben sind die vollständig übersetzte Ausgabe samt Paketbeschreibung (Lokalisierung), die Paketflags (siehe Abschnitt 2.15) und die Debtags (siehe Kapitel 13) zum Paket.

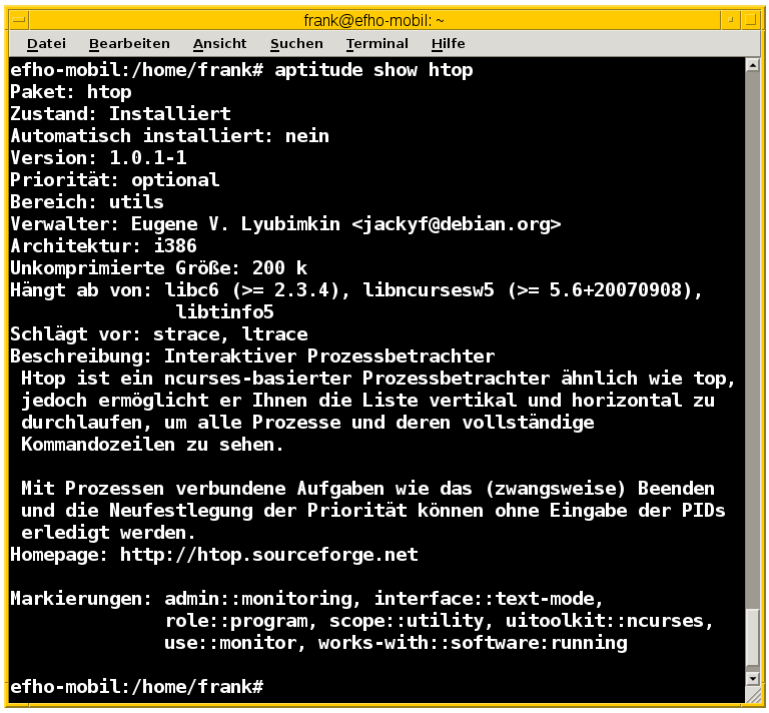

Abbildung 8.1: Ausgabe der Statusinformationen zum Paket *htop* mittels `aptitude`

8.4.6 Anfragen mit `apt-mark`

`apt-mark` ist ebenfalls ein Kommando aus dem Paket *apt*. Es zeigt Ihnen einerseits die Pakete an, die bereits mit einem bestimmten Paketflag (siehe Abschnitt 2.15) versehen wurden, andererseits erlaubt es Ihnen auch, diese Paketflags explizit zu setzen.

Mit den beiden Schaltern `showauto` und `showmanual` zeigen Sie die automatisch bzw. manuell installierten Pakete an. Die nachfolgende Ausgabe zeigt letzteres, auf automatisch installierte Pakete gehen wir in Abschnitt 8.10 genauer ein.

Manuell installierte Pakete anzeigen

```
$ apt-mark showmanual '.*tex$'
dblatex
texlive-xetex
$
```

Für Pakete, deren aktueller Zustand gehalten werden soll, hilft Ihnen dieser Aufruf mit dem Schalter `showhold`. Hier sehen Sie das in Kombination mit den beiden Schaltern `hold` und `unhold` zum Setzen und Entfernen der Markierung am Beispiel des Pakets *xpdf*.

Pakete, deren Zustand gehalten wird

```
# apt-mark hold xpdf
xpdf auf Halten gesetzt.
# apt-mark showhold xpdf
xpdf
# apt-mark unhold xpdf
Halten-Markierung für xpdf entfernt.
#
```

Weiterführende Informationen zu den vier Schaltern `auto`, `manual`, `hold` und `unhold` erhalten Sie unter „Paketflags“ (siehe Abschnitt 2.15), „Festlegen einer Paketversion durch explizites Setzen einer Markierung mit `apt-mark`“ (siehe Kapitel 16) sowie in „Warum ist ein Paket (nicht) installiert“ (siehe Abschnitt 8.15).

8.5 Liste der installierten Pakete anzeigen und deuten

Diese Aktion betrifft häufig nicht nur ein einzelnes Paket, sondern den Gesamtbestand an Software, die auf Ihrem Linuxsystem installiert sind oder waren. Im Alltag ergeben sich eine ganze Reihe von Anwendungsfällen, bei denen diese Aktion durchgeführt wird.

Hintergrund für den *Fall 1* ist der Wunsch nach einem Überblick zum Gesamtzustand eines Systems und vor allem der Software, die darauf installiert ist. Das betrifft insbesondere die Rechnersysteme, die Sie nicht selbst eingerichtet haben und deren Betreuung Ihnen obliegt, bspw. Geräte im Auslieferungszustand oder im Rahmen der Wartung als Bestandteil eines Kundenauftrags. Dabei kommt häufig die Berücksichtigung „gewachsener Infrastruktur“ ins Spiel.

Fall 2 betrifft „großflächige“ Änderungen auf einem Rechnersystem. Sie überprüfen, ob diese korrekt abgelaufen sind. *Fall 3* betrifft die Entwicklung, hier ist die Fehlersuche bei den Paketwerkzeugen zu nennen.

Zu den konkreten Aktionen zählt weiterhin das Herausfinden, ob ein bestimmtes Paket oder Programm auf Ihrem Rechnersystem überhaupt installiert ist, und falls ja, ob dieses vollständig installiert und (bereits) konfiguriert wurde. Wurde es hingegen entfernt, ist zu klären, ob es vollständig entfernt wurde oder noch „Reste“ übrig sind, bspw. paketspezifische Konfigurationsdateien.

Für die Kommandozeile existieren zwei grundlegende Möglichkeiten. Einerseits ist das `dpkg -l` und andererseits `aptitude search ~i`. Die Darstellung in den graphischen Programme klären wir weiter unten genauer.

8.5.1 `dpkg -l` *Paketname* (Langform `--list`)

`dpkg` wird für diese Aufgabe sehr häufig genutzt. Ohne die Angabe des Paketnamens zeigen Sie alle Pakete und Paketreste an, die auf Ihrem System derzeit installiert sind. Geben Sie ein oder mehrere Pakete als Parameter durch Leerzeichen getrennt an, erhalten Sie nur Informationen zu diesen benannten Paketen. In nachfolgender Paketliste zeigen die einzelnen Spalten nacheinander den Paketzustand, den Namen des Pakets (siehe Abschnitt 2.11), dessen Version und die Kurzbeschreibung dazu.

Vollständige Paketliste mit dpkg

```
$ dpkg -l
Gewünscht=Unbekannt/Installieren/R=Entfernen/P=Vollständig Löschen/Halten
| Status=Nicht/Installiert/Config/U=Entpackt/halb konFiguriert/
         Halb installiert/Trigger erWartet/Trigger anhängig
|/ Fehler?=(kein)/R=Neuinstallation notwendig (Status, Fehler: GROSS=schlecht)
||/ Name                                    Version                          ↩
    Beschreibung
+++-=======================================-================================-==========

================================================
ii  a2ps                                    1:4.14-1.1                       GNU a2ps ↩
     - 'Anything to Po
stScript' converter and pretty-printer
rc  bsh                                    2.0b4-12                         Java  ↩
    scripting environment
 (BeanShell) Version 2
...
un  xfce4-icon-theme                       <keine>                          (keine  ↩
    Beschreibung vorhanden)
$
```

Die ersten drei Spalten jeder ausgegebenen Zeile interpretieren Sie anhand der Tabelle 8.1. Die Begriffe in Klammern nennen die englischen Übersetzungen – i.d.R. ist das die Langform der jeweiligen Option. Mit Ausnahme von `W` für *aWaited trigger* entspricht dabei der verwendete Buchstabe in der Spalte jeweils dem Anfangsbuchstaben der Langform der Option.

Tabelle 8.1: Paketzustand deuten

gewünschte Aktion durch den Paketmanager	**Paketstatus**	**Fehler-Schalter**
`u`: Unbekannt (*unknown*)	`n`: Paket ist nicht installiert (*not installed*)	(leer): kein Fehler
`i`: Installieren (*installed*)	`c`: nur die Konfigurationsdatei ist vorhanden (*configured*)	`R`: eine Neuinstallation ist notwendig (*reinstall*)
`h`: Halten (*hold*)	`H`: Paket ist nur halb installiert (*half installed*)	
`r`: Entfernen (*remove*)	`U`: Paket wurde entpackt (*unpacked*)	
`p`: Vollständig Löschen (*purge*)	`F`: Fehlgeschlagene Konfiguration (*failed*)	
	`W`: Trigger erwartet (*aWaited trigger*)	
	`t`: Trigger anhängig (*trigger*)	
	`i`: Installiert (*installed*)	

Kleinbuchstaben zeigen dabei an, dass alles im grünen Bereich ist. Großbuchstaben heben hingegen Fehlerfälle oder auch Zustände hervor, die ihrerseits eine genauere Begutachtung und eine Aktion zu diesem Paket erfordern. Obige Darstellung mit Buchstaben verwendet sowohl `dpkg`, als auch `aptitude`. Tabelle 8.2 zeigt Ihnen die Status-Kombinationen, die Ihnen häufig im Alltag begegnen werden.

Tabelle 8.2: Alltägliche Kombinationen zum Paketzustand

Paketstatus	**Beschreibung**
`ii`	das Paket ist vollständig und fehlerfrei installiert sowie konfiguriert.
`rc`	das Paket wurde gelöscht, aber die Konfigurationsdateien sind noch gespeichert (Abkürzung für *removed, configured, no error*).
`un`	unbekanntes, nicht installiertes Paket (Abkürzung für *unknown, not installed*).
`hi`	das Paket ist installiert, aber auf *hold* gesetzt. Bei Softwareaktualisierungen wird das Paket nicht berücksichtigt und in seinem derzeitigen Zustand belassen (siehe Paketflags in Abschnitt 2.15).

8.5.2 `aptitude search ~i`

`aptitude` kennt dazu das Unterkommando `search` und erwartet danach entweder einen Paketnamen oder ein Flag. In diesem Fall ist es das Flag `~i` für „installierte Pakete“ (Langform `?installed`).

Wie bereits oben genannt, verwendet `aptitude` in der Ausgabe die gleichen Buchstaben wie `dpkg` (siehe Tabelle 8.1). Der Buchstabe `i` bezeichnet ein installiertes Paket, `A` in der dritten Spalte markiert „automatisch installiert“ und deutet auf eine automatisch erfüllte Paketabhängigkeit hin (siehe dazu Abschnitt 8.10). Daneben sehen Sie in der Ausgabe noch den Namen und die Kurzbeschreibung zum jeweiligen Paket.

aptitude listet die installierten LaTeX-Pakete auf

```
$ aptitude search ~i | grep texlive
i   texlive                          - TeX Live: Eine anständige Auswahl der TeX-
i A texlive-base                     - TeX Live: Grundlegende Programme und Datei
i A texlive-bibtex-extra             - TeX Live: Extra BibTeX styles
i A texlive-binaries                 - Binärdateien für TeX Live
i A texlive-common                   - TeX Live: Basiskomponenten
i A texlive-doc-base                 - TeX Live: Dokumentation für TeX Live
$
```

8.5.3 Weitere Möglichkeiten

Graphische Programme wie beispielsweise Synaptic (siehe Abschnitt 6.4.1) und SmartPM (siehe Abschnitt 6.4.3) verwenden keine Buchstaben zur Kennzeichnung des Paketstaus, sondern nutzen stattdessen verschiedenfarbige Kästchen („Icons“). In Abbildung 8.2 sehen Sie alle Möglichkeiten in der vollständigen Übersicht. Installierte Pakete erkennen Sie an der grünen Farbe, weiß/hellgrau kennzeichnet nicht installierte Pakete und rot steht hier für defekte Pakete (Status „broken“).

Abbildung 8.2: Icons zur Darstellung des Paketstatus

8.6 Liste der installierten Kernelpakete anzeigen

Bei dieser Aufgabe hilft Ihnen das Werkzeug `dlocate` [Debian-Paket-dlocate] und zeigt Ihnen an, welche Linux-Kernel und dazugehörigen Pakete auf Ihrem Linuxsystem installiert sind. Es kennt dazu zwei Aufrufvarianten – `dlocate -k` für die Basisinformationen und `dlocate -K` für eine ausführlichere Darstellung.

Die Basisvariante enthält lediglich eine Auflistung der installierten Pakete und zeigt daher nur den Paketnamen (siehe Abschnitt 2.11) an. Die Ausgabe erfolgt dabei ohne Sortierung.

Auflistung der installierten Kernelpakete mit dlocate

```
$ dlocate -k
linux-image-686-pae
linux-headers-3.2.0-4-686-pae
linux-image-2.6-686
linux-image-3.2.0-4-686-pae
linux-headers-3.2.0-4-common
linux-headers-686-pae
$
```

Die ausführliche Darstellung entspricht einer Ausgabe von `dpkg` und zeigt den Installationsstatus, den Paketnamen, die Version und die Paketbeschreibung.

Auflistung der installierten Kernelpakete analog zu dpkg

```
$ dlocate -K
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name                 Version              Description
+++-====================-====================-================================================

ii  linux-headers-3.2.0- 3.2.51-1             i386 Header files for Linux 3.2.0-4-686-pae
ii  linux-headers-3.2.0- 3.2.51-1             i386 Common header files for Linux 3.2.0-4
ii  linux-headers-686-pa 3.2+46               i386 Header files for Linux 686-pae  ↩
    configuration (meta-pack
ii  linux-image-2.6-686  3.2+46               i386 Linux for modern PCs (dummy package)
```

```
ii  linux-image-3.2.0-4- 3.2.51-1             i386 Linux 3.2 for modern PCs
ii  linux-image-686-pae  3.2+46               i386 Linux for modern PCs (meta-package)
$
```

8.7 Liste der installierten, nicht-freien Pakete anzeigen

Mit dem Kommando `vrms` aus dem gleichnamigen Paket *vrms* [Debian-Paket-vrms] erhalten Sie eine Übersicht, welche nicht-freien Pakete auf Ihrem System installiert sind. *vrms* steht dabei als Abkürzung für „Virtual Richard M. Stallman“ und erinnert an den Initiator der GNU-Lizenzen.

Das nachfolgende Beispiel listet die einzelnen Pakete auf, die aus den beiden Bereichen *non-free* und *contrib* ausgewählt wurden, Neben jedem Paket sehen Sie eine Kurzbeschreibung. Die Darstellung entspricht dem Schalter `-e` (Langform `--explain`) und ist seit Debian 8 *Jessie* die Standardeinstellung.

Ausgabe von vrms -e auf einem Desktop-System (Debian 7 Wheezy)

```
$ vrms -e

             Non-free packages installed on efho-mobil

firmware-iwlwifi                    Binary firmware for Intel PRO/Wireless 3945 and 802.11
nautilus-dropbox                    Dropbox integration for Nautilus
openttd-opensfx                     sound set for use with the OpenTTD game
opera                               Fast and secure web browser and Internet suite
skype                               Skype
unrar                               Unarchiver for .rar files (non-free version)

             Contrib packages installed on efho-mobil

flashplugin-nonfree                 Adobe Flash Player - browser plugin

  6 non-free packages, 0.2% of 2696 installed packages.
  1 contrib packages, 0.0% of 2696 installed packages.
$
```

Benötigen Sie hingegen lediglich eine Paketliste ohne zusätzliche Informationen, hilft Ihnen der Schalter `-s` (Langform `--sparse`) weiter. Der Name jedes Pakets wird in einer einzelnen Zeile ausgegeben.

Nach den Paketen aus dem Bereich *non-free* listet `vrms` die *contrib*-Pakete auf. Die Auflistung beider Bereiche wird durch eine schlichte Leerzeile voneinander getrennt und erlaubt somit eine leichte Weiterverarbeitung, bspw. in einem Shellskript.

Ausgabe von vrms -s auf einem Desktop-System (Debian 8 Jessie)

```
$ vrms -s
firmware-iwlwifi
ldraw-parts
skype

flashplugin-nonfree
virtualbox
virtualbox-dkms
virtualbox-guest-dkms
virtualbox-guest-utils
virtualbox-guest-x11
virtualbox-qt
$
```

8.8 Neue Pakete anzeigen

Zu den neuen Paketen zählen die Pakete, die derzeit nicht installiert sind bzw. die noch nie installiert waren. Wurde ein Paket von Ihnen deinstalliert, aber die Konfigurationsdatei blieb erhalten, gilt das entsprechende Paket hingegen nicht mehr als neu. Haben Sie jedoch die Konfigurationsdatei mittels `purge` ebenfalls gelöscht, hat `aptitude` keine Erinnerung mehr an das Paket und betrachtet es wieder als neu.

`aptitude` verfügt mit `~N` (Langform `?new`) über ein Suchmuster, um diese Pakete aufzuspüren. Jede ausgegebene Zeile beginnt mit dem Paketstatus — hier der Kleinbuchstabe `p` zur Kennzeichnung als *neues Paket* --, gefolgt vom Paketnamen und der Kurzbeschreibung zum Paket.

Auflistung aller neuen Pakete mittels aptitude (Auszug)

```
$ aptitude search ~N
p   0ad              - Echtzeit-Strategiespiel über antike Kriegs
p   0ad-data         - Real-time strategy game of ancient warfare
p   0ad-dbg          - Echtzeit-Strategiespiel über antike Kriegs
p   2ping            - Ping-Hilfswerkzeug, um gerichteten Paketve
p   2vcard           - Perl-Skript zur Konvertierung eines Adress
...
$
```

8.9 Pakete nach Prioritäten finden

Wie bereits in „Paket-Priorität und essentielle Pakete“ in Abschnitt 2.13 beschrieben, ist jedem Debianpaket eine bestimmte Wichtigkeit zugeordnet. Dazu zählen erforderlich (*required*), wichtig (*important*), standard (*standard*), optional (*optional*), extra (*extra*) und die Markierung essentiell (*essential*).

Mit `aptitude` können Sie die Paketliste nach diesen Mustern filtern. Dafür kennt das Programm die Suchoption `?priority(wichtigkeit)` bzw. `~pWichtigkeit` als Kurzform. Als `Wichtigkeit` tragen Sie das entsprechende englische Schlüsselwort ein, bspw. `extra` für `?priority(extra)`. Nachfolgend sehen Sie die Pakete, die entsprechend markiert wurden.

Auflistung der extra-Pakete durch aptitude

```
$ aptitude search '?priority(extra)'
p   0ad-dbg          - Echtzeit-Strategiespiel über antike Kriegs
p   389-console      - 389 Management Console
p   4digits          - Zahlenratespiel oder »Bulls and Cows«
p   4store           - Engine zur Datenbankspeicherung und -abfra
p   7kaa-dbg         - Seven Kingdoms Ancient Adversaries - Debug
...
$
```

Essentielle Pakete bezeichnen grundlegende Pakete im Paketbestand. `aptitude` verfügt über eine spezielle Option `~E` (alternativ als Langform `?essential`), um diese essentiellen Pakete anzuzeigen. Dazu rufen Sie es wiederum mit dem Unterkommando `search` auf und erhalten eine Ausgabe auf dem Terminal, die ähnlich zu nachfolgender ist:

Auflistung der essentiellen Pakete durch aptitude

```
$ aptitude search ~E
i A apt              - Paketverwaltung für die Befehlszeile
i   base-files       - Verschiedene Dateien für das Debian-Basiss
i   base-passwd      - Debian Base System Password- und Group-Dat
i   bash             - GNU Bourne Again SHell
i   bsdutils         - Minimalauswahl von Kommandos aus 4.4BSD-Li
i   coreutils        - Grundlegende GNU-Werkzeuge
i A dash             - POSIX-konforme Shell
i   debianutils      - Verschiedene Hilfsprogramme speziell für D
i A diffutils        - Hilfsprogramme zum Dateivergleich
```

```
i   dpkg             - Debian-Paketverwaltungssystem
i   e2fsprogs        - ext2-/ext3-/ext4-Dateisystemwerkzeuge
i   findutils        - Werkzeuge zum Auffinden von Dateien - find
i   grep             - GNU grep, egrep und fgrep
i   gzip             - GNU-Werkzeuge zur Dateikomprimierung
i   hostname         - Werkzeug zum Einrichten und Anzeigen des H
i A libc-bin         - Die »Embedded GNU C Library«: Binärdateien
i   login            - System-Login-Werkzeuge
i   mount            - Tools für das Mounten und die Manipulation
i   ncurses-base     - Beschreibungen gebräuchlicher Terminaltype
i   ncurses-bin      - terminalbezogene Programme und Handbuchsei
i   perl-base        - Minimales Perl-System
i   sed              - Der GNU Streameditor sed
i   sysvinit         - System-V-artige Init-Werkzeuge
i   sysvinit-utils   - System-V-artige Init-Werkzeuge
i   tar              - GNU-Version des tar-Archivierungsprogramms
i   util-linux       - Verschiedene System-Kommandos
$
```

Auffällig an obiger Auflistung ist, dass diese das Paket *apt* enthält, obwohl wir in Abschnitt 2.13 geschrieben haben, dass Sie ein autarkes System auch problemlos ohne APT betreiben können. Der Grund hierfür ist, dass APT sich selbst als essentiell ansieht und sich deswegen selbst nicht deinstallieren will. Da Aptitude zum Teil APTs Bibliotheken benutzt, sieht es diesen Fall genauso[1].

8.10 Automatisch installierte Pakete anzeigen

Darunter fallen Pakete, die automatisch von der Paketverwaltung auf Ihrem System installiert wurden. Das geschieht, wenn Abhängigkeiten von anderen Paketen erfüllt werden müssen. `dpkg` kann diese Information nicht auswerten, jedoch stehen Ihnen mit `apt-mark` und `aptitude` gleich zwei Varianten auf dem Terminal bereit.

8.10.1 `apt-mark` benutzen

Das Paket *apt* [Debian-Paket-apt] beinhaltet das Werkzeug `apt-mark`. Über das Unterkommando `showauto` erhalten Sie alle entsprechend markierten Pakete. Möchten Sie die Liste einschränken, akzeptiert `apt-mark` als weiteren Parameter ein Textfragment des Paketnamens oder einen passenden regulären Ausdruck. Das nachfolgende Beispiel zeigt Ihnen alle automatisch installierten Pakete, deren Paketnamen auf die Zeichenfolge `tex` enden. Das Muster ist als regulärer Ausdruck formuliert.

Automatisch installierte Pakete mit apt-mark anzeigen

```
$ apt-mark showauto '.*tex$'
jadetex
luatex
texlive-luatex
$
```

8.10.2 `aptitude` benutzen

Auf der Kommandozeile suchen Sie im Paketbestand Ihres Linuxsystems mittels `aptitude` über das Unterkommando `search` und dem speziellen Muster `~M` (Langform `?automatic`). Sie erhalten danach eine Liste, die den Paketstatus, den Paketnamen und eine kurze Paketbeschreibung enthält. Alle Pakete sind mit dem Buchstaben `A` für *automatisch installiert* gekennzeichnet.

Automatisch installierte Pakete mit aptitude anzeigen

[1] Bis einschließlich Debian 8 *Jessie* können Sie jedoch das Paket *aptitude* mit Aptitude selbst entfernen. Unter exotischen Umständen hat Aptitude das sogar bereits selbst als Lösungsvorschlag für einen Abhängkeitskonflikt angeboten.

```
$ aptitude search ~M | more
i A a2ps                          - GNU a2ps - »Alles nach PostScript«-Konvert
i A abiword-common                - Effiziente, mächtige Textverarbeitung, unt
i A abiword-plugin-grammar        - Grammatik-Prüfung für AbiWord
i A abiword-plugin-mathview       - Gleichungs-Editor-Erweiterung für AbiWord
i A accountsservice               - Abfragen und Ändern von Informationen von
i A acl                           - Programme für Zugriffskontroll-Listen
...
$
```

Die gleiche Liste erhalten Sie über die Textoberfläche. Dazu filtern Sie die Paketliste mit der Taste **l** und tragen im Suchfeld entweder die Kurzform `~M` oder die Langvariante `?automatic` ein. Das Ergebnis entspricht Abbildung 8.3.

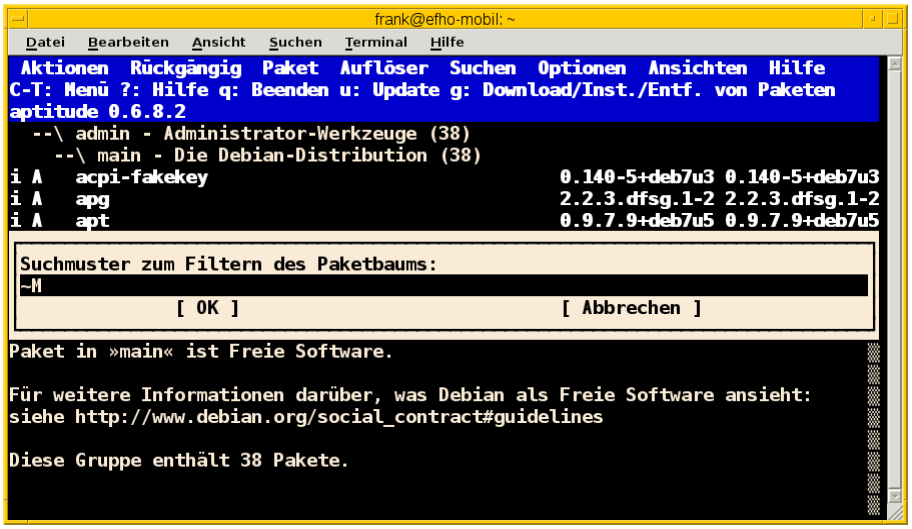

Abbildung 8.3: Ansicht der automatisch installierten Pakete in `aptitude`

8.11 Obsolete Pakete anzeigen

Softwarepakete und insbesondere deren Verfügbarkeit unterliegen einem stetigen Wandel. Im Alltag kommt es ab und zu vor, dass ein Paket, welches Sie auf ihrem System installiert haben, nicht mehr von einem Paketmirror in aktualisierter Form beziehen können. Die Ursachen dafür können sehr vielfältig sein [Hertzog-Obsolete-Packages]:

- Das Paket wurde umbenannt und ist inzwischen unter einem anderen Namen verfügbar. Der Maintainer des Pakets hat die Übergangspakete (siehe „transitional packages“ in Abschnitt 2.7.2) für die aktuelle Veröffentlichung unter dem bisherigen Namen belassen. Später wurden die Übergangspakete entfernt.
- Die letzte verfügbare Version der Software könnte unter einem anderen Namen paketiert worden sein. Entweder waren die Menge der Änderungen so wichtig, dass eine automatische Aktualisierung auf die letzte verfügbare Version nicht bevorzugt wurde, oder einfach weil der Maintainer Ihnen als Benutzer die Möglichkeit geben möchte, mehrere Versionen parallel zu installieren. Ersteres betrifft beispielsweise *request-tracker* und *nagios*, letzteres den Linuxkernel, den Python-Interpreter und viele Bibliotheken.
- Der Entwickler („upstream author“) hat bereits vor längerer Zeit aufgehört, die Software weiter zu warten und niemand anderes hat die Aufgabe von ihm übernommen. Daraufhin hat sich der zuständige Maintainer entschlossen, das Paket aus Debian wieder zu entfernen. Üblicherweise existieren in diesem Fall adäquate Alternativen im Paketbestand.
- Das Paket war seit längerer Zeit in Debian verwaist, niemand hat sich des Pakets angenommen und zusätzlich gab es nur wenige Nutzer. Das Debian-Team zur Qualitätssicherung könnte um dessen Entfernung gebeten haben.

- Das Paket ist ein Kernel-Modul und wurde mit dem Werkzeug `module-assistant` [Debian-Paket-module-assistant] gebaut und installiert[2]
- Das Paket wurde mit `dpkg -i` installiert und war nie über eine APT-Paketquelle verfügbar.

Diese Pakete werden als *obsolete Pakete* bezeichnet und profitieren nicht oder nicht mehr von den regelmäßigen Sicherheitsaktualisierungen. Je länger diese Pakete auf Ihrem System erhalten bleiben, um wahrscheinlicher wird es, dass sich aufgrund der Abhängigkeiten zu anderen Paketen die Aktualisierung ihres gesamten Softwarebestands verzögert und vor allem schwieriger gestaltet. Die Probleme bei der Auflösung von Paketabhängigkeiten nehmen zu.

8.11.1 Recherche auf der Kommandozeile

Mit Hilfe von `aptitude`, dessen Suchfunktion und dem speziellen Muster `~o` (Langform `?obsolete`) spüren Sie diese obsoleten Pakete auf. Trotz des Namens des Suchmusters werden diese Pakete in der Text-Modus-Bedienoberfläche von `aptitude` unter dem Eintrag *Veraltete und selbst erstellte Pakete* (engl.: „Obsolete and Locally Created Packages“) aufgelistet. Dies wird insbesondere dem letzten o.g. Punkt gerecht.

Die nachfolgende Ausgabe umfasst den Paketstatus, den Paketnamen und die entsprechende Kurzbeschreibung zum Paket. Sie sehen dabei auch, dass hierbei für das Paket *pdfstudio* keine Kurzbeschreibung vorliegt und dieses damit nicht die üblichen Qualitätsstandards von Debian erfüllt.

Suche nach obsoleten Paketen mittels aptitude

```
$ aptitude search ~o
i   cupswrapperhl2250dn - Brother HL2250DN CUPS wrapper driver
i   foxitreader         - FoxitReader is a browsing program designed for read
i   gtkdiskfree         - A program to show free/used space on filesystems
i   hl2250dnlpr         - Brother HL-2250DN LPR driver
i   language-env        - simple configuration tool for native language envir
i A libdvdcss2          - Simple foundation for reading DVDs - runtime librar
i A libqt3-mt           - Qt GUI Library (Threaded runtime version), Version
i   odeskteam           - oDesk Team - complete time-logging and verification
i   opera               - Fast and secure web browser and Internet suite
i   pdfedit             - Editor for manipulating PDF documents
i   pdfstudio           -
i   skype               - Wherever you are, wherever they are
i   tpp                 - text presentation program
i   youtube-dl          - downloader of videos from YouTube and other sites
$
```

8.11.2 Recherche in graphischen Programmen

Bei diesen kann u.E. lediglich Synaptic (siehe Abschnitt 6.4.1) die obsoleten Pakete anzeigen. Bei den anderen Programmen fehlt bislang diese Möglichkeit.

Dazu wählen Sie zunächst aus der linken Spalte den Knopf Status aus. Aus der darüberliegenden Liste selektieren Sie danach den Eintrag installiert (lokal oder veraltet). Als Ergebnis erhalten Sie eine Paketliste, welche nur noch die obsoleten Pakete enthält. In Abbildung 8.4 wurden daraus beispielhaft *foxitreader* und *libdvdcss2* markiert.

[2] `module-assistant` war lange Zeit *die* Methode, um Kernel-Module für den aktuell laufenden Kernel zu kompilieren und installieren, die lediglich als Quellcode verfügbar waren. Mittlerweile wurde es größtenteils durch `dkms` (für *Dynamic Kernel Modules Support*, dt.: Dynamische Kernel-Modul-Unterstützung) ersetzt, mit welchem es nicht notwendig ist, lokal `deb`-Pakete generieren und installieren zu lassen.

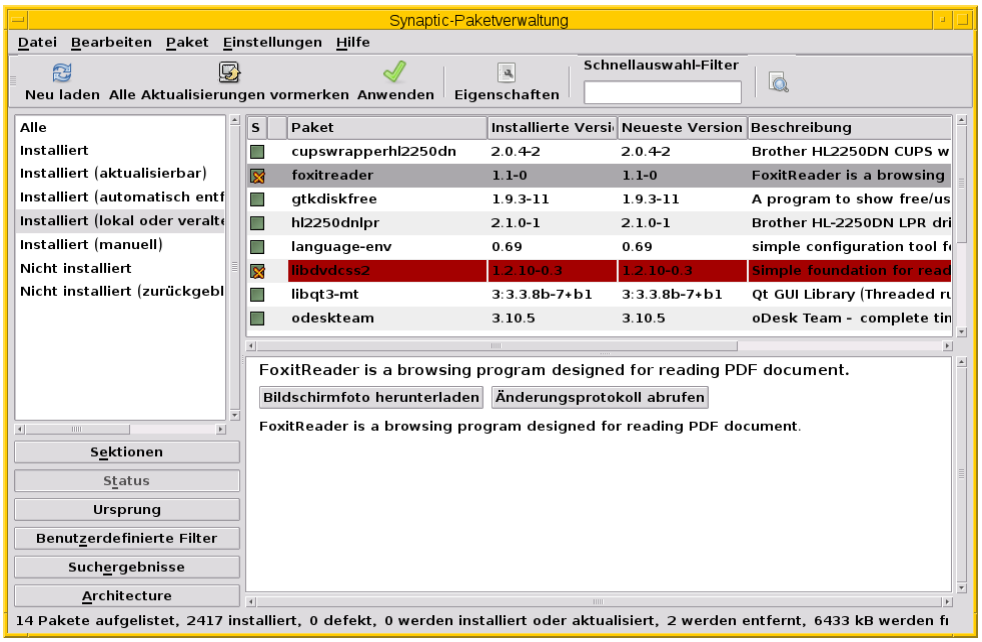

Abbildung 8.4: Ansicht der obsoleten Pakete in Synaptic

8.11.3 Umgang mit diesen Paketen

Ein obsoletes Paket wird aus Sicht der Paketverwaltung wie alle anderen Pakete behandelt und bleibt auf Ihrem Linuxsystem unverändert erhalten, solange dessen Abhängigkeiten nicht verletzt werden. Problematisch ist jedoch die Aktualisierung, da kein Nachfolgepaket existiert. In diesem Fall bestehen nur zwei Möglichkeiten – das Beibehalten der aktuell installierten Version oder der Wechsel auf eine andere, ähnliche Software. Ersteres ist insofern bedenklich, da es die Aktualisierung anderer Pakete über die definierten Paketabhängigkeiten verhindert. Dieser Schritt ist genauso abzuwägen wie der Wechsel zu einer anderen Software, welche vielleicht nicht in allen Punkten ihren Erwartungen und Bedürfnissen entspricht.

8.12 Aktualisierbare Pakete anzeigen

Sowohl APT als auch `aptitude` zeigen Ihnen an, für welche Pakete eine neuere Version bereitsteht. Beide Werkzeuge bieten dafür recht unterschiedliche Parameter und Ausgaben auf dem Terminal.

APT mit dem Kommando `apt-get upgrade -u` (Langform `--show-upgraded`) zeigt Ihnen an, welche Pakete aktualisiert werden. Sie erhalten eine Ausgabe, die der nachfolgenden ähnelt. Die mögliche Option `-s` (Langform `--simulate`) simuliert die Ausführung der Aktualisierung. Letzteres ist nützlich, um zu sehen, was sich ändern wird, wenn Sie das Kommando ausführen.

Anzeige aller Pakete, für die eine neue Version bereitsteht

```
# apt-get upgrade -u -s
Paketlisten werden gelesen...
Abhängigkeitsbaum wird aufgebaut....
Statusinformationen werden eingelesen....
Die folgenden Pakete werden aktualisiert (Upgrade):
  icedove libc-bin libc-dev-bin libc6 libc6-dev libc6-i686 libnss3 libnss3-1d
  linux-headers-3.2.0-4-686-pae linux-headers-3.2.0-4-common
  linux-image-3.2.0-4-686-pae linux-libc-dev virtualbox-guest-source
  virtualbox-ose virtualbox-ose-dkms virtualbox-ose-guest-source
  virtualbox-ose-guest-utils virtualbox-ose-source virtualbox-source
19 aktualisiert, 0 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
```

```
Inst libc-bin [2.13-38+deb7u1] (2.13-38+deb7u4 Debian-Security:7.0/stable [i386]) [libc6: ↩
    i386 ]
Conf libc-bin (2.13-38+deb7u4 Debian-Security:7.0/stable [i386]) [libc6:i386 ]
...
#
```

`aptitude` kennt für diesen Zweck die Suchoption `~U`. Diese steht als Kurzform für `?upgradable`.

Aktualisierbare Pakete mit aptitude anzeigen

```
$ aptitude search ~U
i A cups-common                - Common UNIX Printing System(tm) - gemeinsa
i   iceweasel                  - Webbrowser auf Basis von Firefox
i A libc-bin                   - Die »Embedded GNU C Library«: Binärdateien
i A libc-dev-bin               - Embedded GNU C Library: Entwicklungsbinärd
i   libc6                      - Die »Embedded GNU C Library«: Laufzeitbibl
i A libc6-dev                  - Die »Embedded GNU C Library«: Entwicklungs
...
$
```

Um alle verfügbaren Varianten eines Pakets für alle Veröffentlichungen anzuzeigen, nutzen Sie die `aptitude`-Option `versions`. Nachfolgende Ausgabe illustriert die Recherche nach den Paketen, in denen die Zeichenkette *tzdata* im Paketnamen enthalten ist. Hier werden zudem ausschließlich Pakete aus der Veröffentlichung *stable* bezogen. Die Sortierung erfolgt paketweise, d.h. zunächst erhalten Sie eine Zeile mit dem Paketnamen und darunter zusätzliche Informationen zur verfügbaren Version. Die erste Spalte zeigt dabei den Paketstatus an, danach die Versionsnummer, die Veröffentlichung und als letztes die Priorität (siehe dazu „Veröffentlichungen mischen“ in Kapitel 20).

Die verfügbaren Versionen zu den Paketen tzdata anzeigen

```
aptitude versions 'tzdata'
Paket tzdata:
p   2015f-0+deb8u1             stable  500
i   2015g-0+deb8u1             stable- 500

Paket tzdata-java:
p A 2015f-0+deb8u1             stable  500
i A 2015g-0+deb8u1             stable- 500
$
```

Wünschen Sie nur eine kompakte Ausgabe zu einem konkreten Paket ohne Darstellung des Paketnamens, helfen Ihnen die beiden Schalter `--show-package-names` mit dem Wert `never` und `--group-by` mit dem Wert `none` weiter. Ersteres blendet den Paketnamen aus, während der zweite Schalter die Gruppierung in der Ausgabe deaktiviert. Ausführlicher gehen wir auf die Gruppierung in Abschnitt 10.7 ein.

Kompakte Ausgabe ohne Paketname

```
$ aptitude versions 'tzdata-java' --show-package-names=never --group-by=none
p A 2015f-0+deb8u1             stable  500
i A 2015g-0+deb8u1             stable- 500
$
```

Als gleichwertige Alternative steht Ihnen auch das zusätzliche Werkzeug `apt-show-versions` aus dem gleichnamigen Debianpaket zur Verfügung [Debian-Paket-apt-show-versions] (siehe auch „Aus welchem Repo kommen die Pakete“ in Abschnitt 8.18). Die nachfolgende Ausgabe zeigt den Status des Pakets `base-files` an. Daraus erkennen Sie, daß für dieses ein neueres Paket bereitsteht.

Kompakte Ausgabe mittels apt-show-versions

```
$ apt-show-versions base-files
base-files:amd64/jessie 8+deb8u2 upgradeable to 8+deb8u3
$
```

Bei den graphischen Programmen zur Paketverwaltung kann lediglich Synaptic (siehe Abschnitt 6.4.1) die aktualisierbaren Pakete anzeigen. Dazu wählen Sie zunächst den Knopf `Benutzerdefinierte Filter` aus der linken Spalte aus. Aus der darüberliegenden Liste selektieren Sie danach den Eintrag `Aktualisierbar (Upstream)`. Als Ergebnis erhalten Sie eine Paketliste, welche nur noch die Pakete enthält, die erneuerbar sind (siehe Abbildung 8.5).

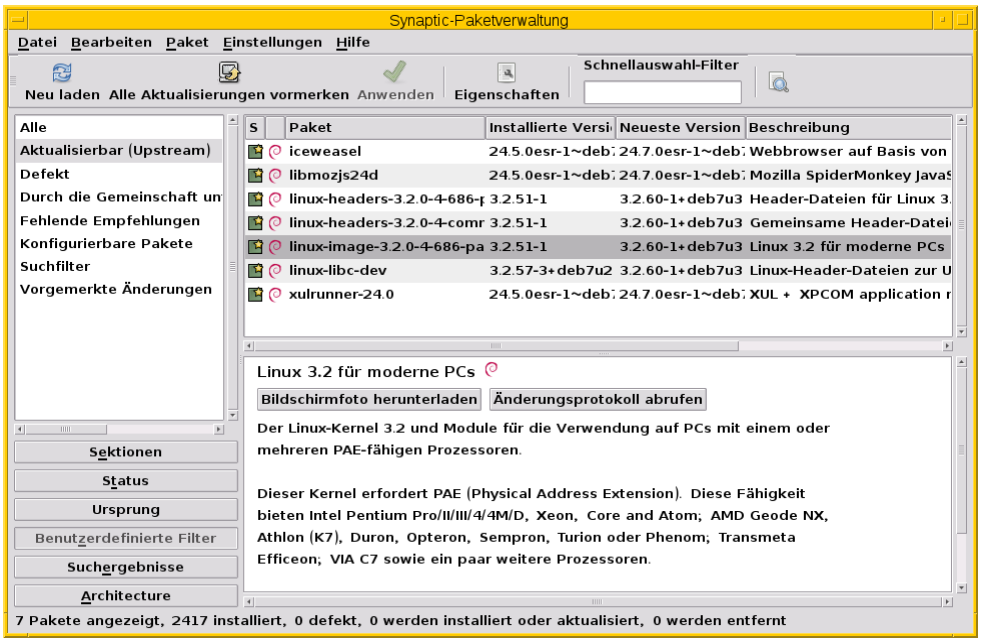

Abbildung 8.5: Ansicht der aktualisierbaren Pakete in Synaptic

8.13 Installationsgröße eines Pakets

Die Frage, wieviel Platz ein installiertes Paket auf dem Speichermedium belegt, beantworten Sie am besten mit dem Aufruf `dlocate -du Paketname` [Debian-Paket-dlocate]. Als Ergebnis erhalten Sie eine Auflistung mit einer Datei pro Zeile, bei der die jeweilige Größe in der ersten Spalte in Kilobyte angegeben ist. Die letzte Zeile summiert die Einzelwerte auf. Nachfolgendes Listing zeigt den Aufruf für das Paket *htop*.

Ermittlung der Installationsgröße des Pakets htop mit dlocate

```
$ dlocate -du htop
136 /usr/bin/htop
4   /usr/share/applications/htop.desktop
4   /usr/share/doc/htop/AUTHORS
8   /usr/share/doc/htop/changelog.Debian.gz
8   /usr/share/doc/htop/changelog.gz
4   /usr/share/doc/htop/copyright
4   /usr/share/doc/htop/README
4   /usr/share/man/man1/htop.1.gz
4   /usr/share/menu/htop
4   /usr/share/pixmaps/htop.png
180 insgesamt
$
```

Möchten Sie den benötigten Speicherplatz bereits vor der Installation wissen, hilft Ihnen `aptitude` weiter. Mit Hilfe des zusätzlichen Schalters `-Z` ergänzt `aptitude` die Größenangabe hinter jedem Paket, hier beispielhaft für *aptitude-doc-it* und *aptitude-doc-ru*. Wird ein + vor der Zahl dargestellt, wird dieser Speicherplatz zur Installation des Paketes benötigt, ein − vor der Zahl gibt hingegen den frei werdenden Platz an, wenn das Paket entfernt wird. Die Gesamtsumme der Änderungen erfahren Sie in der Statusnachricht darunter.

Ausgabe der Installationsgröße eines Pakets mit aptitude

```
# aptitude -Z install aptitude-doc-it aptitude-doc-ru
Die folgenden NEUEN Pakete werden zusätzlich installiert:
   aptitude-doc-it <+1.082 kB>  aptitude-doc-ru <+1.408 kB>
   0 Pakete aktualisiert, 2 zusätzlich installiert, 0 werden entfernt
   und 14 nicht aktualisiert.
   636 kB an Archiven müssen heruntergeladen werden. Nach dem Entpacken
   werden 2.490 kB zusätzlich belegt sein.
...
#
```

8.14 Größtes installiertes Paket finden

8.14.1 dpigs

Mit dem Programm `dpigs`[3] aus dem Paket *debian-goodies* [Debian-Paket-debian-goodies] zeigen Sie die Programme bzw. Programmpakete an, die den meisten Speicherplatz auf ihrem Debiansystem belegen. Es wertet dazu die Ausgabe des Kommandos `grep-status` aus dem Paket *dctrl-tools* [Debian-Paket-dctrl-tools] aus. Ausführlicher besprechen wir die Werkzeuge im Abschnitt Schlagworte verwenden (Debtags) (siehe Abschnitt 13.6)].

Ein Aufruf ohne weitere Parameter listet Ihnen die zehn größten Speicherfresser auf ihrem System auf. Dabei enthält die erste Spalte die Größe in Kilobyte und die zweite Spalte den Paketnamen.

Ausgabe von dpigs

```
$ dpigs
399871 texlive-fonts-extra
377071 texlive-latex-extra-doc
129158 libreoffice-core
91551 pdfstudio
88963 libgl1-mesa-dri
86913 texlive-lang-greek
86446 texlive-pstricks-doc
80298 libwine
80178 openjdk-6-jre-headless
80175 linux-image-3.2.0-4-686-pae
$
```

`dpigs` verfügt nur über einige wenige, aber durchaus nützliche Schalter. Um beispielsweise die Anzahl der ausgegebenen Pakete zu begrenzen, nutzen Sie die Option `-n` (Langform `--lines=N`), wobei `n` bzw. `N` ein Zahlenwert für die gewünschte Anzahl ist. Der Schalter `-S` zeigt die Pakete an, die hingegen die größten Quelldaten haben. Mit dem Schalter `-H` rechnet `dpigs` die Größenangaben in menschlich lesbare Werte um. Der Schalter `-H` steht hierbei als Abkürzung für *human readable*. Kombinieren Sie die drei genannten Schalter, sieht die Recherche nach den fünf großen Paketen auf einem Desktopsystem in etwa folgendermaßen aus:

Ausgabe von dpigs mit Einschränkung auf die fünf Spitzenreiter in menschenlesbarer Form

```
$ dpigs -S --lines=5 -H
   1.0G texlive-extra
 299.7M libreoffice
 274.7M texlive-base
 131.4M chromium-browser
 120.5M calligra
$
```

[3] „pig“ ist Englisch für Schwein bzw. Sau. Es geht sozusagen um Debianpakete, die den Plattenplatz versauen, auch bekannt als „Plattenplatzschweine“.

8.14.2 aptitude

`aptitude` kann mit dem o.g. vorgestellten Programm `dpigs` problemlos mithalten. Es macht nur ein wenig mehr Mühe, die gewünschten Informationen zu erhalten.

Mit dieser Schrittfolge erhalten Sie eine entsprechend sortierte Liste in aufsteigender Reihenfolge in der Text-Modus-Oberfläche:

1. Mit Ctrl-t oder **F10** öffnen Sie die Menüleiste.
2. Dort wählen Sie den Eintrag Ansichten → Neue einfache Paketansicht aus.
3. Darin schränken Sie nun die Ansicht auf die installierten Pakete ein. Dazu drücken Sie **l** (für „limit“) und geben als Filter `~i` für „nur installierte Pakete“ ein. Den Vorgang schließen Sie mit der Eingabetaste ab.
4. Nun fehlt nur noch die passende Sortierung. Diese erhalten Sie durch das Drücken von **S** und der Eingabe von `install size`. Mit der Eingabetaste schließen Sie den Vorgang ab.
5. Jetzt springen Sie mit der **Ende**-Taste ans Ende der Liste und sehen die schlimmsten Plattenplatzverbraucher ihres Systems.

Im Terminal finden Sie die zehn Pakete mit dem meisten Plattenplatzverbrauch durch eine Kombination von `aptitude` und dem Standard-UNIX-Kommando `tail`. An `aptitude` übergeben Sie dabei neben dem Unterkommando `search` mehrere Optionen. Während `--sort installsize` für die Sortierung nach dem Paketattribut „belegter Plattenplatzverbrauch“ sorgt, filtert `'~i'` nur alle installierten Pakete aus der Liste heraus. Die Ausgabe enthält den Paketstatus, den Namen des Pakets und die Paketbeschreibung in Kurzform (siehe Abschnitt 8.5). Der anschließende Aufruf des Kommandos `tail` ohne weitere Optionen beschränkt die Darstellung auf die letzten zehn Zeilen der Ausgabe von `aptitude` und somit die zehn größten Pakete.

Suche nach den größten installierten Paketen mit aptitude (Namensliste)

```
$ aptitude search --sort installsize '~i' | tail
i A cube2-data                - demo game and content for the Cube2 engine
i A nexuiz-data               - Nexuiz game data files
i A torcs-data-tracks         - data files for TORCS game - Tracks set
i A supertuxkart-data         - 3D kart racing game (data)
i A flightgear-data-aircrafts - FlightGear Flight Simulator -- standard ai
i A flightgear-data-ai        - FlightGear Flight Simulator -- standard AI
i A nexuiz-textures           - Textures for Nexuiz
i A sauerbraten               - 3D first-person shooter game
i A flightgear-data-base      - FlightGear Flight Simulator -- base files
i A 0ad-data                  - Real-time strategy game of ancient warfare
$
```

Möchten Sie zudem wissen, wieviel Platz die einzelnen Pakete verbrauchen, hilft Ihnen der Schalter `-F` (Langform `--display-format`) gefolgt von einem Formatstring weiter. Darüber steuern Sie die Ausgabe des Suchergebnisses von `aptitude`. Mit einem Formatstring legen Sie die Informationen und deren Reihenfolge fest, in der diese ausgegeben werden sollen (siehe `aptitude`-Formatstrings in Abschnitt 10.4).

In unserem Fall benötigen wir lediglich den Plattenplatzverbrauch und den Paketnamen. Die beiden Spalteninhalte spezifizieren Sie über die beiden Formatbezeichner `%I` für die Installationsgröße (engl. „installed size“) und `%p` für den Paketnamen (engl. „package name“). Nachfolgende Darstellung ist aufsteigend, d.h. das zehntkleinste Paket sehen Sie in der ersten und das größte in der letzten Zeile der Auflistung.

Suche nach den größten Paketen mit aptitude (Größe und Paketname), aufsteigende Sortierung

```
$ aptitude search -F '%I %p' --sort installsize '~i' | tail
272 MB   cube2-data
278 MB   nexuiz-data
302 MB   torcs-data-tracks
306 MB   supertuxkart-data
320 MB   flightgear-data-aircrafts
466 MB   flightgear-data-ai
523 MB   nexuiz-textures
652 MB   sauerbraten
751 MB   flightgear-data-base
```

```
1359 MB  0ad-data
$
```

Für eine umgekehrte, absteigende Darstellung kommt noch das hilfreiche UNIX-Kommando `tac` ins Spiel. Über eine Pipe leiten Sie die Ausgabe von `tail` and `tac` weiter. Dieses dreht die Ausgabe um, sodass die letzte Zeile zuerst ausgegeben wird, danach die vorletzte u.s.w. Die nachfolgende Ausgabe zeigt eine Auflistung der fünf größten Pakete in absteigender Reihenfolge. Da `tail` ohne Parameter stets 10 Zeilen ausgibt, wurde dessen Aufruf noch um die Angabe `-5` ergänzt.

Suche nach den größten Paketen mit aptitude (Größe und Paketname), absteigende Sortierung

```
$ aptitude search -F '%I %p' --sort installsize '~i' | tail -5 | tac
1359 MB  0ad-data
751 MB   flightgear-data-base
652 MB   sauerbraten
523 MB   nexuiz-textures
466 MB   flightgear-data-ai
$
```

8.15 Warum ist ein Paket installiert

Mit der Zeit sammeln sich auf Ihrem System recht viele Pakete an. Bedingt durch die vorab definierten Paketabhängigkeiten schaufelt die Paketverwaltung etliches an Daten auf die Platte und füllt den verfügbaren Speicherplatz stetig, aber gnadenlos.

Möchten Sie klären, warum ein Paket installiert wurde, gibt Ihnen `aptitude` dazu bereitwillig Auskunft. Es versteht dazu das Unterkommando `why`, mit dem es Ihnen die Gründe auflistet, die zur Installation des besagten Pakets geführt haben. Grundlage dafür sind einerseits die Paketflags (siehe dazu Abschnitt 2.15) und andererseits die Einträge in der Paketbeschreibung. `aptitude` wertet daraus die fünf Felder `Recommends`, `Conflicts`, `Depends`, `Replaces` und `Provides` aus (siehe Abschnitt 4.1). Über diese vorgenannten Felder legt der Paketmaintainer die Beziehungen zu anderen Paketen fest.

Nachfolgendes Beispiel zeigt den Aufruf zum Paket *xpdf*. Ersichtlich wird daraus, dass das Paket *texpower* wiederum *xpdf* oder ein Paket aus der Gruppe *pdf-viewer* empfiehlt. Die Alternative wird hier durch einen senkrechten Strich – das Pipe-Symbol – dargestellt. Das Paket *xpdf* ist bereits installiert, wird jedoch wiederum auch vom Paket *texlive-latex-extra* genutzt. Das Paket *texpower* wurde automatisch installiert, was Sie anhand des Buchstabens `A` in der dritten Spalte der dritten Zeile in nachfolgender Ausgabe sehen können.

Ausgabe von aptitude zur Klärung der Installation (Variante 1)

```
$ aptitude why xpdf
i   texlive-latex-extra Empfiehlt texpower (>= 0.2-2)
i A texpower            Empfiehlt xpdf | pdf-viewer
$
```

Für eine wesentlich ausführlichere Ausgabe nutzen Sie den Schalter `-v`. Dieser besitzt die übliche Langform `--verbose` und ergänzt die Ausgabe um alle Pakete, die zur Installation führen können. Hintergrund dafür ist, dass `aptitude` aus den Paketinformationen sämtliche Abhängigkeits- und Installationspfade errechnet und Ihnen diese auflistet.

Ausführliche Darstellung der Installationspfade (Ausschnitt)

```
$ aptitude --verbose why xpdf
i   mc Schlägt vor xpdf | pdf-viewer

i   nautilus Schlägt vor evince | pdf-viewer
i   xpdf     Liefert     pdf-viewer

i   dot2tex          Empfiehlt    pgf (>= 2.00) | texlive-pstricks
i A texlive-pstricks Hängt ab von texlive-base (>= 2012.20120516)
i A texlive-base     Schlägt vor  xpdf-reader | pdf-viewer
i   xpdf             Liefert      pdf-viewer

...
$
```

In Kombination mit dem Schalter `--show-summary` fassen Sie diese Informationen zusammen und erhalten eine kompaktere Darstellung. Der Schalter erlaubt die fünf Werte `no-summary`, `first-package`, `first-package-and-type`, `all-packages` und `all-packages-with-dep-versions`. Geben Sie den Schalter beim Aufruf nicht an, wird `show-summary` auf den Wert `no-summary` gesetzt. Spezifizieren Sie keinen Wert für `show-summary`, wird hingegen `first-package` angenommen. Damit erhalten Sie bei einem Aufruf die folgende Ausgabe – hier beispielhaft für das Paket *foomatic-db*:

Zusammenfassung für das Paket foomatic-db

```
$ aptitude -v --show-summary why foomatic-db
Pakete, die foomatic-db erfordern:
  bluetooth
  cups-pdf
  printer-driver-gutenprint
$
```

Der Wert `first-package-and-type` liefert Ihnen zudem eine Begründung für die Abhängigkeit:

Zusammenfassung für das Paket foomatic-db (mit Begründung)

```
$ aptitude -v --show-summary=first-package-and-type why foomatic-db
Pakete, die foomatic-db erfordern:
  [Hängt ab von] bluetooth
  [Hängt ab von] cups-pdf
  [Hängt ab von] printer-driver-gutenprint
$
```

Eine Übersicht zu allen Paketpfaden, die zum genannten Paket führen, erhalten Sie mit Hilfe des Wertes `all-packages`. Dabei stehen die Buchstaben hinter den Paketnamen für `Recommends` (`R`), `Conflicts` (`C`), `Depends` (`D`), `Replaces` (**?**) und `Provides` (`P`).

Zusammenfassung für das Paket foomatic-db (ausführlicher)

```
$ aptitude -v --show-summary=all-packages why foomatic-db
Pakete, die foomatic-db erfordern:
  bluetooth E: bluez-cups H: cups E: foomatic-filters E: foomatic-db-engine E: foomatic-db
  cups-pdf H: cups E: foomatic-filters E: foomatic-db-engine E: foomatic-db
  printer-driver-gutenprint H: cups E: foomatic-filters E: foomatic-db-engine E: foomatic- ↩
      db
$
```

Benötigen Sie zudem noch die Versionsnummer, von der dieses Paket abhängt, setzen Sie den Wert von `--show-summary` auf den Wert `all-packages-with-dep-versions`:

Zusammenfassung für das Paket foomatic-db (ausführlicher mit Versionsangabe)

```
$ aptitude -v --show-summary=all-packages-with-dep-versions why foomatic-db
Pakete, die foomatic-db erfordern:
  bluetooth E: bluez-cups H: cups E: foomatic-filters (>= 4.0) E: foomatic-db-engine (>=  ↩
      4.0) E: foomatic-db
  cups-pdf H: cups E: foomatic-filters (>= 4.0) E: foomatic-db-engine (>= 4.0) E: foomatic- ↩
      db
  printer-driver-gutenprint H: cups (>= 1.3.0) E: foomatic-filters (>= 4.0) E: foomatic-db- ↩
      engine (>= 4.0) E: foomatic-db
$
```

Bestehen hingegen keine Gründe für eine Installation oder `aptitude` kann diese nicht zweifelsfrei ermitteln, liefert es die nachfolgende Ausgabe – hier am Beispiel des Pakets *libruby-extras*:

Ausgabe von aptitude zur Klärung der Installation (Variante 2)

```
$ aptitude why libruby-extras
Kann keinen Grund für die Installation von libruby-extras finden.
$
```

Das Gegenstück zum Unterkommando `why` ist `why-not`. `aptitude` listet damit den Grund auf, warum das Paket *bislang nicht* installiert oder *bereits wieder entfernt* wurde. Nachfolgendes Beispiel zeigt das anhand des Audioprogramms `mplayer`. Dieses Paket wird von *rtmpdump* vorgeschlagen, welches automatisch installiert wurde und vom Paket *youtube-dl* empfohlen wird. Außerdem kollidieren die beiden Pakete *mplayer* und *mplayer2* miteinander, sodass im Bedarfsfall nur eines von beiden auf ihrem System installiert sein darf.

Ausgabe von aptitude zur Klärung der Installation (Variante 3)

```
$ aptitude why-not mplayer
i   youtube-dl Empfiehlt      rtmpdump
i A rtmpdump   Schlägt vor    mplayer
p   mplayer2   Liefert        mplayer
p   mplayer2   Kollidiert mit mplayer
$
```

Kann `aptitude` keinen Grund für die Entfernung finden, meldet es sich mit einer kurzen Meldung dazu zurück – hier am Beispiel des KDE Desktop Managers aus dem Paket *kdm*:

Ausgabe von aptitude zur Klärung der Installation (Variante 4)

```
$ aptitude why-not kdm
Kann keinen Grund für die Entfernung von kdm finden.
$
```

8.16 Liste der zuletzt installierten Pakete anzeigen

Als Verantwortlicher für Ihr Linuxsystem möchten Sie ab und an wissen, was zuletzt auf dem System passiert ist. Dazu zählt insbesondere das Auswerten der Logdateien und beinhaltet die Änderungen der Paketliste – was wurde installiert, gelöscht bzw. aktualisiert etc. `dpkg` erfasst alle Änderungen im Paketbestand in der Logdatei `/var/log/dpkg.log`. Ältere `dpkg`-Logdateien werden vom Programm `logrotate` nach dem Rotationsprinzip archiviert und irgendwann auch komprimiert. Die zweit- und drittjüngsten Logdateien finden Sie beispielsweise in den beiden Dateien `/var/log/dpkg.log.1` und `/var/log/dpkg.log.2.gz` wieder. Nachfolgender Auszug zeigt die Suche in der aktuellen Logdatei von `dpkg` mittels `grep` anhand des Schlüsselwortes „install“.

Recherche in den Logdateien von dpkg mittels grep

```
$ grep " install " /var/log/dpkg.log
2014-08-06 15:12:19 install texlive-games:all <keine> 2012.20120611-2
2014-08-06 15:59:34 install qxw:i386 <keine> 20110923-1
2014-08-08 10:46:42 install games-thumbnails:all <keine> 20120227
2014-08-08 10:46:43 install goplay:i386 <keine> 0.5-1.1
2014-08-08 19:42:14 install ocrad:i386 <keine> 0.22~rc1-2
$
```

Reicht Ihnen dieses Ergebnis noch nicht aus, suchen Sie zusätzlich noch in allen wegrotierten Logdateien. Zur Anwendung kommt hier das Kommando `zcat` aus dem Paket *gzip*, welches ein *essentielles* Paket darstellt. Dies ist auf einer Standard-Installation der Fall. Es gibt jedoch auch noch eine alternative Implementation im Paket `zutils` [Debian-Paket-zutils].

Untenstehende Ausgabe wurde zusätzlich mittels `sort` aufsteigend nach Datum sortiert, d.h. die untersten Einträge enthalten die jüngsten Änderungen im Paketbestand. Mit dem UNIX-Kommando `tail` beschränken Sie die Ausgabe des Rechercheergebnisses auf lediglich zehn Einträge.

Recherche in den Logdateien von dpkg mittels zcat

```
$ zcat -f /var/log/dpkg.log* | grep " install " | sort | tail
2014-07-23 20:18:35 install libparse-debianchangelog-perl:all <keine> 1.2.0-1
2014-07-23 20:18:36 install libxml-simple-perl:all <keine> 2.20-1
2014-07-23 20:18:36 install patchutils:i386 <keine> 0.3.2-1.1
2014-07-23 20:18:37 install lintian:all <keine> 2.5.10.4
2014-07-26 16:03:02 install libapt-pkg-doc:all <keine> 0.9.7.9+deb7u2
```

```
2014-08-06 15:12:19 install texlive-games:all <keine> 2012.20120611-2
2014-08-06 15:59:34 install qxw:i386 <keine> 20110923-1
2014-08-08 10:46:42 install games-thumbnails:all <keine> 20120227
2014-08-08 10:46:43 install goplay:i386 <keine> 0.5-1.1
2014-08-08 19:42:14 install ocrad:i386 <keine> 0.22~rc1-2
$
```

Die hier verwendete Option `-f` benötigen Sie, damit `zcat` auch nicht-komprimierte Dateien – in diesem Fall `/var/log/dpkg.log` – ausgibt.[4]

Alternativ können Sie auch `zgrep` verwenden, das spart ein klein wenig Tipparbeit. Damit die Sortierung gelingt, muss dort allerdings die Ausgabe von vorangestellten Dateinamen mit der Option `-h` unterdrückt werden:

Recherche in den Logdateien von dpkg mittels zgrep

```
$ zgrep -h " install " /var/log/dpkg.log* | sort | tail -7
2014-07-23 20:18:37 install lintian:all <keine> 2.5.10.4
2014-07-26 16:03:02 install libapt-pkg-doc:all <keine> 0.9.7.9+deb7u2
2014-08-06 15:12:19 install texlive-games:all <keine> 2012.20120611-2
2014-08-06 15:59:34 install qxw:i386 <keine> 20110923-1
2014-08-08 10:46:42 install games-thumbnails:all <keine> 20120227
2014-08-08 10:46:43 install goplay:i386 <keine> 0.5-1.1
2014-08-08 19:42:14 install ocrad:i386 <keine> 0.22~rc1-2
$
```

8.17 Paketabhängigkeiten anzeigen

Wie in der Einführung zum Buch bereits genannt, besteht Debian aus einer riesigen Menge einzelner Pakete. Dabei werden größere, komplexe Funktionalitäten in kleine, separate Bausteine zerlegt. Diese Bausteine sind als einzelne `deb`-Pakete verfügbar, die häufig einander bedingen. In der Paketbeschreibung jedes `deb`-Pakets ist der Bezug zu den anderen Paketen hinterlegt. Genaueres dazu lesen Sie unter Debian-Paketformat im Detail in Kapitel 4.

Vor der Veränderung des Paketbestands – d.h. dem Hinzufügen, Entfernen und Aktualisieren eines oder mehrerer Pakete – prüfen APT und `aptitude`, ob diese Abhängigkeiten nach den Änderungen weiterhin erfüllt sind. Die Abhängigkeiten müssen erfüllt sein, damit Ihr Linuxsystem weiterhin funktioniert und dieses für Sie benutzbar bleibt. Sind die Abhängigkeiten jedoch nicht erfüllt, werden Ihnen die Pakete und die erforderlichen Paketoperationen angezeigt, die zur Aufrechterhaltung eines sauberen Paketbestands durchzuführen sind.

8.17.1 Paketabhängigkeiten auflisten

Dazu stehen Ihnen einerseits das Werkzeug `apt-cache` mit den beiden Unterkommandos `show` und `depends` zur Verfügung, andererseits auch `aptitude` mit speziellen Suchoptionen. Als weitere Information benötigen beide Programme noch den Namen des Pakets, zu dem es die Paketabhängigkeiten ausgeben sollen.

Die beiden nachfolgenden Ausgaben illustrieren die Wirkung zum Paket *xpdf*. Die erste Ausgabe greift auf `apt-cache show` in Verbindung mit dem UNIX-Kommando `grep` und einer Pipe zurück, um aus der Paketbeschreibung die entsprechende Zeile mit den Abhängigkeiten herauszufiltern.

Suche mittels grep in der Ausgabe von apt-cache

```
$ apt-cache show xpdf | grep Depends
Depends: lesstif2 (>= 1:0.94.4), libc6 (>= 2.4), libgcc1 (>= 1:4.1.1), libpoppler19 (>=  ↩
    0.18.4), libstdc++6 (>= 4.1.1), libx11-6, libxt6
$
```

In der zweiten Darstellung sehen Sie den Einsatz des Unterkommandos `depends`. Jedes benötigte Debianpaket wird in einer einzelnen Zeile in alphabetisch aufsteigender Reihenfolge und ohne die Versionsnummer ausgegeben.

Detaillierte Ausgabe der Paketabhängigkeiten

[4] Ist die alternative `zcat`-Implementation aus dem Paket *zutils* installiert, ist die Option `-f` nicht mehr erforderlich.

```
$ apt-cache depends xpdf
xpdf
  Hängt ab von: lesstif2
  Hängt ab von: libc6
  Hängt ab von: libgcc1
  Hängt ab von: libpoppler19
  Hängt ab von: libstdc++6
  Hängt ab von: libx11-6
  Hängt ab von: libxt6
  Empfiehlt: poppler-utils
  Empfiehlt: poppler-data
  Empfiehlt: gsfonts-x11
  Empfiehlt: cups-bsd
$
```

Ohne weitere Optionen enthält die Ausgabe alle Abhängigkeiten und Empfehlungen zu dem Paket. Mit den folgenden Optionen spezifizieren Sie die Ausgabe noch genauer.

`-i` (Langform `--important`)
Einschränkung auf die wichtigen Abhängigkeiten, d.h. nur unerfüllte (*unmet*) und direkte Abhängigkeiten (*depends*)

`--no-pre-depends`
blendet die Pakete aus, die vorher installiert sein müssen

`--no-depends`
direkte Abhängigkeiten ausblenden

`--no-recommends`
die Empfehlungen für weitere Pakete ausblenden

`--no-suggests`
Angaben für Vorschläge werden unterdrückt

`--no-conflicts`
blendet die Pakete aus, die mit dem Paket in Konflikt stehen, d.h. nicht gleichzeitig installiert sein dürfen

`--no-breaks`
blendet die Pakete aus, die das Paket funktionsunfähig machen

`--no-replaces`
Pakete, die das aktuelle Paket ersetzen, werden nicht angezeigt

`--no-enhances`
Pakete, die das aktuelle Paket erweitern, werden nicht angezeigt

`--installed`
begrenzt die Ausgabe nur auf die installierten Pakete

Eine Anwendung der Optionen zeigt die nachfolgende Ausgabe, hier für das Paket *xfce4* und mit der Option `--no-depends`. Angezeigt werden nur die vorgeschlagenen und empfohlenen Pakete zu *xfce4*.

Ausgabe der vorgeschlagenen und empfohlenen Pakete zu xfce4

```
$ apt-cache depends xfce4 --no-depends
xfce4
  Schlägt vor: xfprint4
  Schlägt vor: xfce4-goodies
  Empfiehlt: xorg
  Empfiehlt: desktop-base
  Empfiehlt: thunar-volman
  Empfiehlt: tango-icon-theme
  Empfiehlt: xfce4-notifyd
$
```

Wie oben schon angerissen, versteht `aptitude` eine Reihe von speziellen Suchmustern. Eines davon ist `~Dmuster` als Abkürzung für *depends*, welches Sie mit dem Unterkommando `search` kombinieren. *muster* bezeichnet hier den Namen oder das Textfragment eines Pakets. Um beispielsweise alle Pakete aufzulisten, die eine Abhängigkeit auf das Paket *xpdf* in der Paketbeschreibung deklariert haben, nutzen Sie das Kommando `aptitude search ~Dxpdf`. Das Ergebnis ist eine mehrspaltige Auflistung der Pakete mit deren Installationsstatus, Paketnamen und Kurzbeschreibung (siehe dazu Liste der installierten Pakete anzeigen und deuten in Abschnitt 8.5).

Ausgabe der Paketabhängigkeiten mit aptitude

```
$ aptitude search ~Dxpdf
p   eficas                - Graphical editor for Code Aster command files
p   impressive            - Werkzeug zur Präsentation von PDF-Dateien mit
p   muttprint-manual      - Handbuch für muttprint
p   page-crunch           - PDF and PS manipulation for printing needs
p   wiipdf                - Präsentiert eine PDF-Datei mittels Wiimote
$
```

8.17.2 Anzeige der umgekehrten Paketabhängigkeiten

Diese Aktivität übersetzen Sie mit der Frage „Welche anderen Pakete benötigt Paket *x*?“, auch genannt *Rückwärtsabhängigkeit*. Zur Beantwortung der Frage helfen Ihnen einerseits wiederum `apt-cache` mit dem Unterkommando `rdepends`, andererseits das Kommando `apt-rdepends` aus dem gleichnamigen Paket *apt-rdepends* [Debian-Paket-apt-rdepends] und auch `aptitude` selbst weiter.

Ausgabe der umgekehrten Paketabhängigkeiten mit apt-cache für das Paket xfce4

```
$ apt-cache rdepends xfce4
xfce4
Reverse Depends:
  xfwm4
  task-xfce-desktop
 |desktop-base
  education-desktop-xfce
$
```

Pakete, die von weiteren Paketen abhängen, sind in der Ausgabe von `apt-cache` mit einem senkrechten Strich („Pipe“) gekennzeichnet. Deutlicher wird `apt-rdepends`, da es die Abhängigkeiten noch weitaus stärker auflöst. Nachfolgende Darstellung zeigt daher nur einen Ausschnitt.

Ausgabe der umgekehrten Paketabhängigkeiten mit apt-rdepends (Ausschnitt)

```
$ apt-rdepends xfce4 | more
Reading package lists... Done
Building dependency tree
Reading state information... Done
xfce4
  Depends: gtk2-engines-xfce (>= 2.8.0)
  Depends: orage (>= 4.8.0)
  Depends: thunar (>= 1.2.0)
  Depends: xfce4-appfinder (>= 4.8.0)
  Depends: xfce4-mixer (>= 4.8.0)
  Depends: xfce4-panel (>= 4.8.0)
  Depends: xfce4-session (>= 4.8.0)
  Depends: xfce4-settings (>= 4.8.0)
  Depends: xfce4-utils (>= 4.8.0)
  Depends: xfconf (>= 4.8.0)
  Depends: xfdesktop4 (>= 4.8.0)
  Depends: xfwm4 (>= 4.8.0)
gtk2-engines-xfce
  Depends: libatk1.0-0 (>= 1.12.4)
  Depends: libc6 (>= 2.3.6-6~)
```

```
...
$
```

Das Ergebnis von `apt-rdepends` wird vielleicht leichter verständlich, wenn Sie die Paketabhängigkeiten graphisch darstellen. Dabei hilft Ihnen das Programm `dotty` aus dem Paket *graphviz* [Graphviz]. Für das Paket *tcpdump* sieht der Aufruf wie folgt aus.

Erzeugung der Abhängigkeiten als Dot-Datei

```
$ apt-rdepends -d tcpdump | dot > tcpdump.dot
Reading package lists... Done
Building dependency tree
Reading state information... Done
$
```

Das Ergebnis der von `apt-rdepends` zu `dot` weitergeleiteten und in der Datei `tcpdump.dot` abgespeicherten Relationsmenge zeigen Sie mit dem Programm `dotty` an (siehe Abbildung 8.6).

Aufruf von dotty

```
$ dotty tcpdump.dot
$
```

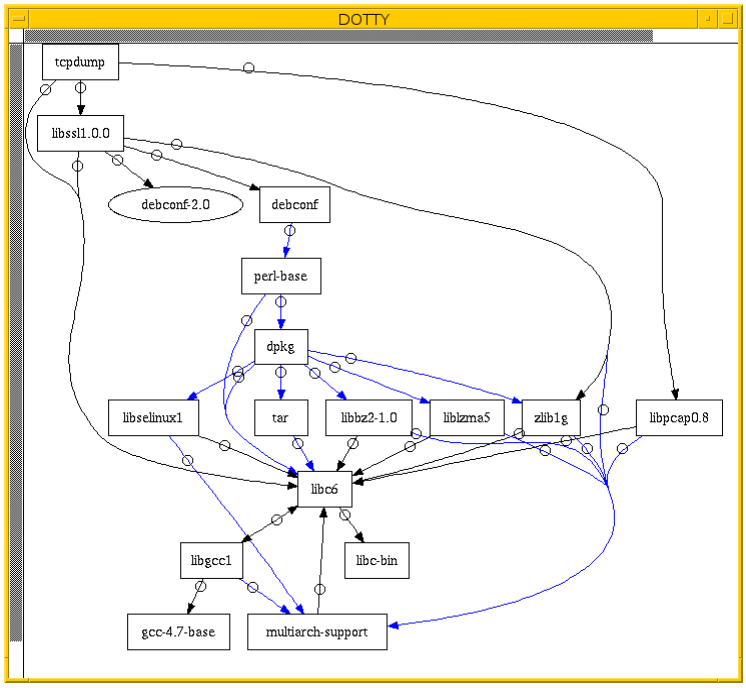

Abbildung 8.6: Darstellung der umgekehrten Paketabhängigkeiten mit `dotty`

Zur Suche nach umgekehrten Paketabhängigkeiten hilft Ihnen `aptitude` mit dem Suchmuster `~Rmuster` als Abkürzung für *reverse-depends*. Dieses Suchmuster kombinieren Sie wieder mit dem Unterkommando `search` und dem Namen oder Textfragment eines Pakets. Für das Paket *xfce4* erhalten Sie nachfolgende Ausgabe:

Ausgabe der umgekehrten Paketabhängigkeiten mit aptitude (Ausschnitt)

```
$ aptitude search ~Rxfce4
p   aterm                - Afterstep XVT - ein VT102 Emulatur für das
p   aterm-ml             - Afterstep XVT - ein VT102-Emulatur für das
i   dpkg                 - Debian-Paketverwaltungssystem
p   dunst                - Minimalistischer Benachrichtigungs-Daemon
```

```
p   eterm                - Enlightened Terminal Emulator
p   evilvte              - Leichtgewichtiger Terminal-Emulator auf Ba
i A exo-utils            - Werkzeugdateien für libex
p   exo-utils-dbg        - Debuginformationen für exo-utils
i   gnome-terminal       - GNOME-Terminalemulator
i A gstreamer0.10-alsa   - GStreamer-Erweiterung für ALSA
...
$
```

Möchten Sie hingegen nur die Pakete anzeigen, die sich gegenseitig direkt bedingen, hilft Ihnen `apt-rdepends` mit der Option `-r`. Nachfolgend zeigt es die definierte Abhängigkeit zwischen den beiden Paketen *xfce4* und *task-xfce-desktop*.

Ausgabe sich gegenseitig bedingender Pakete mit apt-rdepends

```
$ apt-rdepends xfce4 -r
Reading package lists... Done
Building dependency tree
Reading state information... Done
xfce4
  Reverse Depends: task-xfce-desktop (3.14.1)
task-xfce-desktop
$
```

8.17.3 Prüfen, ob die Abhängigkeiten des gesamten Systems erfüllt sind

APT liefert über das Werkzeug `apt-get` und dessen Unterkommando `check` ein kleines Diagnosewerkzeug mit. Es aktualisiert den Paketzwischenspeicher (siehe Kapitel 7) und prüft, ob auf Ihrem Linuxsystem beschädigte Abhängigkeiten vorliegen. Das beinhaltet alle installierten Pakete sowie die bereits entpackten, aber noch nicht konfigurierten Pakete [Debian-Anwenderhandbuch-apt-optionen].

Prüfung auf beschädigte Abhängigkeiten mit apt-get check

```
# apt-get check
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
#
```

8.17.4 Zusammenfassung aller unerfüllten Abhängigkeiten im Paketcache

Das Werkzeug `apt-cache` zeigt Ihnen eine Zusammenfassung aller unerfüllten Abhängigkeiten im Paketzwischenspeicher (siehe Kapitel 7). Dazu bietet es das Unterkommando `unmet`, welches Sie auch noch um einen Paketnamen bzw. eine Liste davon ergänzen können. Die dargestellte Liste zeigt die Funktionalität zum Paket *wireshark* und beinhaltet auch die nicht installierten Vorschläge der Pakete.

Auflistung aller unerfüllten Abhängigkeiten für Pakete, die mit wireshark beginnen

```
$ apt-cache unmet wireshark*
Paket wireshark Version 1.8.2-5wheezy10 hat eine unerfüllte Abhängigkeit:
 Ersetzt: ethereal (< 1.0.0-3)
Paket libwireshark2 Version 1.8.2-5wheezy10 hat eine unerfüllte Abhängigkeit:
 Ersetzt: wireshark-common (< 1.4.0~rc2-1)
Paket libwireshark-data Version 1.8.2-5wheezy10 hat eine unerfüllte Abhängigkeit:
 Ersetzt: wireshark-common (< 1.4.0~rc2-1)
Paket wireshark-common Version 1.8.2-5wheezy10 hat eine unerfüllte Abhängigkeit:
 Ersetzt: ethereal-common (< 1.0.0-3)
Paket libwireshark-dev Version 1.8.2-5wheezy10 hat eine unerfüllte Abhängigkeit:
 Ersetzt: wireshark-dev (< 1.4.0~rc2-1)
Paket wireshark-dev Version 1.8.2-5wheezy10 hat eine unerfüllte Abhängigkeit:
```

```
 Ersetzt: ethereal-dev (< 1.0.0-3)
frank@efho-mobil:~$ apt-cache unmet wireshark
Paket wireshark Version 1.8.2-5wheezy10 hat eine unerfüllte Abhängigkeit:
 Ersetzt: ethereal (< 1.0.0-3)
$
```

8.18 Aus welchem Repo kommen die Pakete

Nutzen Sie Pakete aus verschiedenen Paketquellen in `/etc/apt/sources.list` (siehe Abschnitt 3.3), ist es hilfreich zu wissen, woher APT ein Paket bei der Installation oder Aktualisierung entnehmen würde. Bei der Beantwortung dieser Frage helfen Ihnen die Programme `apt-cache`, `apt-show-versions`, `apt` und `aptitude` weiter – aber jedes auf seine Art.

8.18.1 Paketquelle untersuchen mit `apt-cache policy`

Rufen Sie `apt-cache` lediglich mit dem Schalter `policy` und *ohne* die weitere Angabe eines Pakets auf, untersucht das Programm jede einzelne Paketquelle, die Sie in `/etc/apt/sources.list` eingetragen haben. Das Ergebnis der Analyse ist zweispaltig. In der linken Spalte erscheint ein Zahlenwert zur Priorität des jeweiligen Eintrags, wie er von `apt-pinning` genutzt wird (siehe dazu Kapitel 22). In der rechten Spalte sehen Sie die Paketquelle anhand der heruntergeladenen, lokalen Paketliste mit zusätzlichen Informationen wie bspw. der Veröffentlichung oder des Distributionsbereichs. Nachfolgende Darstellung zeigt die Ausgabe für eine Standardinstallation von Debian *Wheezy* in der Version 7.5 mit dem Nutzungsschwerpunkt Deutschland.

Bewertung der Paketquellen mit apt-cache policy

```
$ apt-cache policy
Paketdateien:
 100 /var/lib/dpkg/status
     release a=now
 500 http://security.debian.org/ wheezy/updates/non-free Translation-en
 500 http://security.debian.org/ wheezy/updates/main Translation-en
 500 http://security.debian.org/ wheezy/updates/contrib Translation-en
 500 http://security.debian.org/ wheezy/updates/non-free i386 Packages
     release v=7.0,o=Debian,a=stable,n=wheezy,l=Debian-Security,c=non-free
     origin security.debian.org
 500 http://security.debian.org/ wheezy/updates/contrib i386 Packages
     release v=7.0,o=Debian,a=stable,n=wheezy,l=Debian-Security,c=contrib
     origin security.debian.org
 500 http://security.debian.org/ wheezy/updates/main i386 Packages
     release v=7.0,o=Debian,a=stable,n=wheezy,l=Debian-Security,c=main
     origin security.debian.org
 500 http://ftp.de.debian.org/debian/ wheezy/non-free Translation-en
 500 http://ftp.de.debian.org/debian/ wheezy/main Translation-en
 500 http://ftp.de.debian.org/debian/ wheezy/main Translation-de_DE
 500 http://ftp.de.debian.org/debian/ wheezy/main Translation-de
 500 http://ftp.de.debian.org/debian/ wheezy/contrib Translation-en
 500 http://ftp.de.debian.org/debian/ wheezy/non-free i386 Packages
     release v=7.5,o=Debian,a=stable,n=wheezy,l=Debian,c=non-free
     origin ftp.de.debian.org
 500 http://ftp.de.debian.org/debian/ wheezy/contrib i386 Packages
     release v=7.5,o=Debian,a=stable,n=wheezy,l=Debian,c=contrib
     origin ftp.de.debian.org
 500 http://ftp.de.debian.org/debian/ wheezy/main i386 Packages
     release v=7.5,o=Debian,a=stable,n=wheezy,l=Debian,c=main
     origin ftp.de.debian.org
Mit Pinning verwaltete Pakete:
$
```

Geben Sie hingegen beim Aufruf als Parameter einen Paketnamen an, prüft `apt-cache`, ob das Paket bereits auf Ihrem System installiert ist oder ob es ein neueres Paket gibt und falls ja, von welchem Paketmirror das Paket in diesem Fall käme.

Beispiel 1 zeigt das Vorgehen anhand des Pakets *gdm3*. Im vorliegenden Fall ist dieses bereits installiert (Status von `dpkg`) Falls es das noch nicht wäre, käme das Paket aus dem deutschen Debian-Repository.

Verfügbarkeit für das Paket gdm3 feststellen

```
$ apt-cache policy gdm3
gdm3:
  Installiert:           3.4.1-8
  Installationskandidat: 3.4.1-8
  Versionstabelle:
 *** 3.4.1-8 0
        500 http://ftp.de.debian.org/debian/ wheezy/main i386 Packages
        100 /var/lib/dpkg/status
$
```

Beispiel 2 betrifft das Paket *linux-libc-dev*. Dieses ist bereits in Version 3.2.51-1 installiert, aber es gibt eine aktuellere Variante (3.2.57-3) sowie zusätzlich eine Sicherheitsaktualisierung (Security-Update) mit der Versionsnummer 3.2.46-1+deb7u1. In diesem Fall ist die Version 3.2.57-3 der Installationskandidat, da dieses Paket die aktuellste Variante darstellt.

Verfügbarkeit für das Paket linux-libc-dev feststellen

```
$ apt-cache policy linux-libc-dev
linux-libc-dev:
  Installiert:           3.2.51-1
  Installationskandidat: 3.2.57-3
  Versionstabelle:
     3.2.57-3 0
        500 http://ftp.de.debian.org/debian/ wheezy/main i386 Packages
 *** 3.2.51-1 0
        100 /var/lib/dpkg/status
     3.2.46-1+deb7u1 0
        500 http://security.debian.org/ wheezy/updates/main i386 Packages
$
```

Als **Beispiel 3** steht das Paket *kteatime* im Fokus. Dieses ist noch nicht installiert und könnte nachgezogen werden. Dabei käme das Paket aus dem deutschen Debian-Repository.

Verfügbarkeit für das Paket kteatime feststellen

```
$ apt-cache policy kteatime
kteatime:
  Installiert:           (keine)
  Installationskandidat: 4:4.8.4-1
  Versionstabelle:
     4:4.8.4-1 0
        500 http://ftp.de.debian.org/debian/ wheezy/main i386 Packages
$
```

8.18.2 Verfügbare Paketversionen mit `apt-cache madison` ermitteln

`apt-cache` verfügt ebenso über ein Unterkommando namens `madison`. Damit finden Sie heraus, welche Pakete derzeit von den Spiegelservern in einer neueren Version verfügbar sind. Nachfolgender Ausdruck zeigt das für das Paket *apt-doc*.

Ausgabe von apt-cache mit dem Unterkommando madison für apt-doc

```
$ apt-cache madison apt-doc
   apt-doc | 0.9.7.9+deb7u2 | http://ftp.de.debian.org/debian/ wheezy/main i386 Packages
   apt-doc | 0.9.7.9+deb7u2 | http://security.debian.org/ wheezy/updates/main i386 Packages
       apt | 0.9.7.9+deb7u2 | http://ftp.de.debian.org/debian/ wheezy/main Sources
```

```
       apt | 0.9.7.9+deb7u2 | http://security.debian.org/ wheezy/updates/main Sources
$
```

Wollen Sie diese Informationen nicht nur für die auf Ihrem System genutzten Architekturen und Veröffentlichungen sehen, sondern für alle Architekturen und Veröffentlichungen von Debian, so können Sie das Programm `rmadison` aus dem Paket *devscripts* [Debian-Paket-devscripts] verwenden. Es fragt dazu per HTTP eine regelmäßig aktualisierte Quelle im Internet ab, d.h. Sie brauchen Internetzugriff, um `rmadison` zu nutzen.

Als Parameter erwartet `rmadison` einen oder mehrere Paketnamen, nach denen es dann recherchiert. Als Ergebnis sehen Sie in der linken Spalte den Paketnamen, gefolgt von der Versionsnummer des Pakets, der Veröffentlichung und am Schluss die Architektur, für die das Paket verfügbar ist. Im nachfolgenden Beispielaufruf ist die Architektur *all*, da es sich um das Dokumentationspaket *apt-doc* handelt, welches auf allen Plattformen gleich ist.

Ausgabe von rmadison für apt-doc

```
$ rmadison apt-doc
 apt-doc | 0.8.10.3+squeeze1 | squeeze          | all
 apt-doc | 0.8.10.3+squeeze2 | squeeze-lts      | all
 apt-doc | 0.9.7.9+deb7u2    | wheezy-security | all
 apt-doc | 0.9.7.9+deb7u2    | wheezy           | all
 apt-doc | 1.0.6             | jessie           | all
 apt-doc | 1.0.6             | sid              | all
 apt-doc | 1.1~exp1          | experimental     | all
 apt-doc | 1.1~exp2          | experimental     | all
$
```

Sowohl das `apt-cache`-Unterkommando `madison` als auch `rmadison` sind benannt nach `madison`, einem Programm welches beim Verwalten der Debian-APT-Archive zum Einsatz kommt. Da das originale `madison` aber direkt auf dem Server aufgerufen werden muss, auf dem das Debian-APT-Archiv zusammengebaut wurde, ist es nur für Debian-Entwickler nutzbar – und selbst für diese eher umständlich, da sie sich erst per SSH auf jenem Server einloggen müssen. Diese beiden Ersatzkommandos sind da doch wesentlich einfacher und vor allem für jedermann zu benutzen.

8.18.3 Verfügbare Paketversionen mit `apt-show-versions` ermitteln

apt-show-versions [Debian-Paket-apt-show-versions] ist ein Paket, welches jedoch nicht zur Standardinstallation von Debian zählt. Als Parameter geben Sie dem gleichnamigen Programm den Namen des gewünschten Pakets an, für welches Sie Informationen zu den verfügbaren Varianten wünschen. Das Programm ist insbesondere in gemischten Umgebungen interessant, d.h. wenn Pakete aus verschiedenen Veröffentlichungen gleichzeitig verwendet werden.

Nachfolgendes Beispiel zeigt die Ausgabe für das Paket *zsh* auf einem Debian 7 *Wheezy*. Hier ist kein neueres Paket verfügbar:

Versionen anzeigen für das Paket zsh

```
$ apt-show-versions zsh
zsh/wheezy uptodate 4.3.17-1
$
```

Anders als bei dem obigen Beispiel ist es für das Paket *vlc* unter Debian 8 *Jessie*, für das ein aktuelleres Paket bereitsteht:

Versionen anzeigen für das Paket vlc

```
$ apt-show-versions vlc
vlc:amd64/jessie 2.2.0~rc2-2+deb8u1 upgradeable to 1:2.2.1-dmo3
$
```

8.18.4 APT 1.0 mit dem Unterkommando `list`

Ab Debian 8 *Jessie* und Ubuntu 14.04 LTS *Trusty Tahr* wird APT auf der Basis der Version 1.0 ausgeliefert. Ab dieser Version verlieren die zuvor vorgestellten Kommandos `apt-cache` und `apt-show-versions` etwas an Bedeutung, da das Kommando `apt` mit dem neuen Unterkommando `list` und einem Paketnamen als Parameter ähnliches leistet. Die Ausgabe des Paketnamens auf dem Terminal erfolgt in Farbe, nur lässt sich das hier im Buch leider weniger gut illustrieren.

Nachfolgend sehen Sie wiederum die Ausgabe für das Paket *zsh*. Nach dem Paketnamen erscheinen die Veröffentlichung, die Versionsnummer und die genutzte Architektur (hier *amd64*). In den Klammern sehen Sie neben dem Installationsstatus eine kurze Angabe, auf welche Version Sie das Paket aktualisieren können.

Neue Möglichkeiten mit APT 1.0

```
$ apt list zsh
Listing... Done
zsh/unstable,testing,now 5.0.5-3 amd64 [installed,upgradable to: 5.0.5-4]
$
```

8.18.5 Aktualisierbare Pakete mit `aptitude` ermitteln

Auch `aptitude` hilft Ihnen dabei, die Pakete zu finden, für die es Neuerungen gibt. Näheres dazu lesen Sie unter Aktualisierbare Pakete anzeigen in Abschnitt 8.12.

8.19 Pakete über den Namen finden

Diese Suchvariante ist der häufigste Weg zur Recherche nach gewünschten Paketen. Alle Werkzeuge zur Paketverwaltung bieten eine entsprechende Suchfunktion an, variieren dabei jedoch stark in der Form sowie der Menge der Möglichkeiten. Namentlich ähnliche Pakete sind mit `dpkg`, `apt-cache` und `aptitude` über ein Unterkommando und ein Muster auffindbar. `dpkg` hat ein eigenes Musterformat, `apt-cache` und `aptitude` unterstützen hingegen Reguläre Ausdrücke. Bei den graphischen Programmen ist die Suche über Muster bislang nicht oder lediglich eingeschränkt implementiert.

8.19.1 `dpkg`

`dpkg` sucht nur in der Paketliste. Dazu bietet es die Option `-l` und listet darüber nur derzeit oder früher schon einmal installierte Pakete auf. Es erwartet als weiteren Parameter entweder einen vollständigen Paketnamen oder einen Suchausdruck mit Platzhalter (engl. *Wildcard*). Bitte schließen Sie den Suchausdruck in einfache oder doppelte Hochkommata ein, damit die Shell nicht versucht, den Platzhalter selbst auszuwerten.

Fahndung nach den Paketen für xpdf mittels dpkg

```
$ dpkg -l 'xpdf*'
Gewünscht=Unbekannt/Installieren/R=Entfernen/P=Vollständig Löschen/Halten
| Status=Nicht/Installiert/Config/U=Entpackt/halb konFiguriert/
         Halb installiert/Trigger erWartet/Trigger anhängig
|/ Fehler?=(kein)/R=Neuinstallation notwendig (Status, Fehler: GROSS=schlecht)
||/ Name           Version      Architektur  Beschreibung
+++-==============-============-============-=================================
ii  xpdf           3.03-10      amd64        Portable Document Format (PDF) re
un  xpdf-reader    <keine>                   (keine Beschreibung vorhanden)
un  xpdf-utils     <keine>                   (keine Beschreibung vorhanden)
$
```

Obiger Ausgabe entnehmen Sie, dass nur das Paket *xpdf* installiert ist. Die beiden anderen Pakete namens *xpdf-reader* und *xpdf-utils* waren schon einmal installiert und sind daher `dpkg` bereits bekannt. Deswegen erscheint als Paketstatus die Buchstabenfolge `un` für „nicht mehr installiert“.

Ausgabeformat des Paketstatus
Das Ausgabeformat sowie die Buchstaben am Zeilenanfang erklären wir ausführlicher in den beiden Abschnitten `dpkg` und Liste der installierten Pakete anzeigen und deuten (siehe Abschnitt 8.5).

8.19.2 APT

Das Kommando `apt-cache` liefert in der Standardeinstellung das umfangreichste Suchergebnis auf der Kommandozeile. Ohne weitere Suchoptionen durchsucht `apt-cache search` die Paketliste und erstöbert den Paketnamen, die Paketabhängigkeiten sowie die kurze als auch die ausführliche Paketbeschreibung.

`apt-cache` erwartet nach dem Unterkommando `search` ein Textfragment als Suchbegriff. Nachfolgend sehen Sie das Suchergebnis zum Paket *lintian* [Debian-Paket-lintian]. Es zeigt das gefundene Paket sowie dessen Kurzbeschreibung dazu an.

Suche nach einem Paket

```
$ apt-cache search lintian
dput - Debian package upload tool
elida - pbuilder mail interface
fixincludes - Fix non-ANSI header files
libdebian-package-html-perl - generates HTML from a Debian source/binary package
debaux - Debian-Hilfsprogramme
devscripts - Skripte, die das Leben eines Debian-Paketbetreuers erleichtern
lintian - Debian-Paketprüfer
$
```

8.19.3 aptitude

`aptitude` berücksichtigt bei der Suche üblicherweise nur den Paketnamen. Es sucht jedoch auf Wunsch auch in der Paketliste und der Paketbeschreibung. Beinhaltet der Paketname eine Tilde (`~`) oder ein Fragezeichen (`?`), wird der Paketname als Suchmuster aufgefasst. In Folge werden alle Pakete berücksichtigt, die diesem Suchmuster entsprechen. Dazu füttern Sie `aptitude` mit den folgenden Optionen:

`~dsuchbegriff` (Langform `?description(suchbegriff)`)
: Suche nach dem *suchbegriff* in der Paketbeschreibung.

`~nsuchbegriff` (Langform `?name(suchbegriff)`)
: Suche nach *suchbegriff* im Namen des Pakets. *suchbegriff* wird hier als Teilzeichenkette betrachtet und findet bspw. bei `apt` die Pakete *apt*, *aptitude* und *synaptic*.

`?exact-name(suchbegriff)`
: Suche nach Paketen, deren Paketnamen exakt mit dem Suchbegriff übereinstimmen.

`?term(suchbegriff)`
: Volltextsuche nach *suchbegriff* im Namen und der Beschreibung des Pakets.

`?term-prefix(suchbegriff)`
: Volltextsuche nach Begriffen, die den *suchbegriff* als Präfix beinhalten.

`?user-tag(suchbegriff)`
: Suche nach Begriffen, die den *suchbegriff* als benutzerdefinierte Markierung beinhalten.

Der Kommandozeilenaufruf von `aptitude` ist ähnlich zu `apt-cache` – auch hier folgt auf das Unterkommando `search` die Suchoption oder nur das Textfragment zur Recherche. Beispielhaft interessierte uns das Paket *xpdf*. Wie Sie der nachfolgenden Ausgabe entnehmen können, sucht `aptitude` per Default nur in den Paketnamen. Das Suchergebnis zeigt, dass insgesamt drei Pakete verfügbar sind, die *xpdf* im Namen tragen. Davon ist nur das erste auf dem System installiert.

Suche nach xpdf mittels aptitude

```
$ aptitude search xpdf
i   xpdf          - Leseprogramm für das Portable Document Format (PDF)
p   xpdf-reader   - Übergangspaket für xpdf
p   xpdf-utils
$
```

In der Textoberfläche von `aptitude` begrenzen Sie zunächst die Auswahl mit der Taste **l**. In das Eingabefeld tragen Sie den Suchtext oder das o.g. Suchmuster ein. Daraufhin erscheinen in der Übersicht alle Pakete, die diesem Muster entsprechen (siehe Abbildung 8.7).

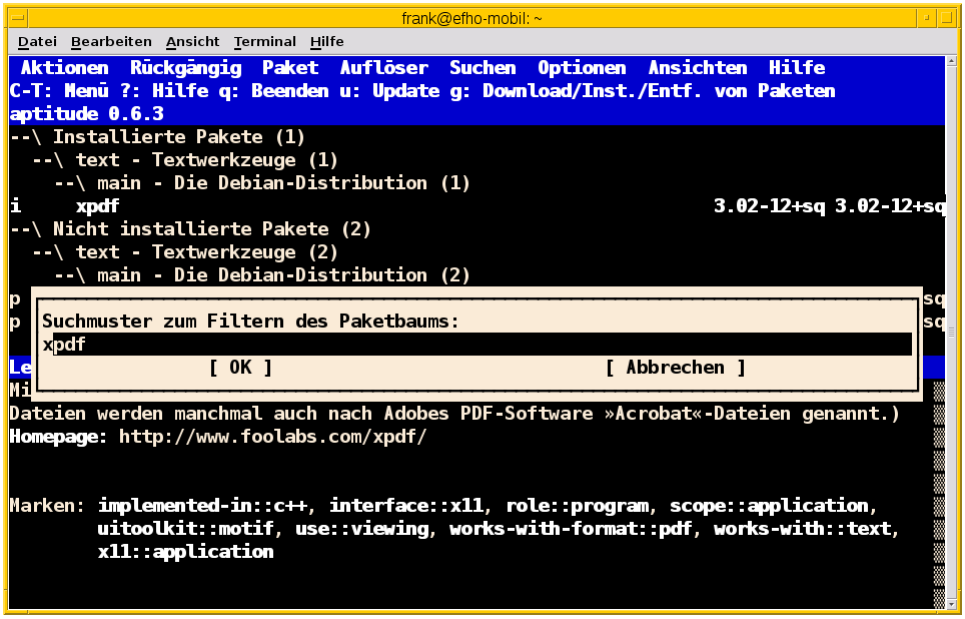

Abbildung 8.7: Ergebnis der Limitierung nach `xpdf` über die Aptitude-TUI

8.19.4 Synaptic

Synaptic aus Abschnitt 6.4.1 bietet Ihnen zwei Varianten zur Suche an – einerseits eine Schnellsuche und andererseits eine ausführliche Suche. Die *Schnellsuche* verbirgt sich hinter dem Suchfenster oben rechts und ist mit dem Begriff Schnellauswahl-Filter betitelt. Geben Sie dort einen Text ein, durchsucht Synaptic die Paketliste und filtert nur die heraus, deren Paketname mit dem Text beginnen. Dabei werden Platzhalter und Reguläre Ausdrücke nicht unterstützt. Als Ergebnis wird die dargestellte Paketliste auf die gefundenen Pakete eingeschränkt.

Die *ausführlichere Suche* erreichen Sie mittels Ctrl-F oder alternativ über den Knopf mit der Lupe. Es öffnet sich ein Fenster mit einem Eingabefeld für den Suchtext. Darunter befindet sich ein Auswahlfeld, wo Sie die Suche nach Name, Beschreibung und Name, Betreuer (siehe auch Abschnitt 8.21), Version, Abhängigkeiten und den bereitgestellten Paketen spezifizieren können. Bei dieser Suche versteht Synaptic auch Fragmente, d.h. es interpretiert den Suchtext als Teilstring. Abbildung 8.8 zeigt das Suchergebnis für das Fragment `fce`.

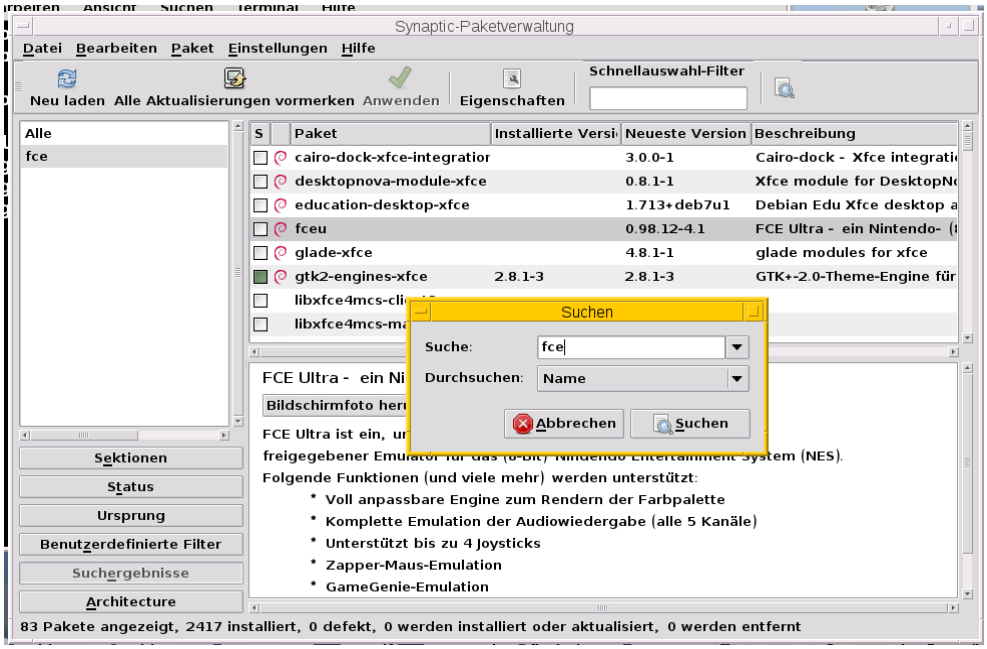

Abbildung 8.8: Ergebnis der Suche nach dem Fragment `fce` in Synaptic

8.19.5 SmartPM

SmartPM (Abschnitt 6.4.3) besitzt nur eine einfachere Suchfunktion. Diese ist als Suchfeld in die graphische Bedienoberfläche integriert. Das Suchfeld erreichen Sie ebenfalls über die Tastenkombination Ctrl-F. SmartPM versteht auch Fragmente, d.h. es interpretiert den Suchtext als Teilstring, sucht bislang jedoch nur im Paketnamen.

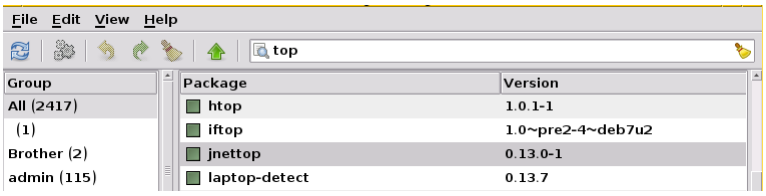

Abbildung 8.9: Ergebnis der Suche nach dem Fragment `top` in SmartPM

8.19.6 Suche über die Webseite des Debian-Projekts

Nicht nur für den Einstieg, sondern auch für den Alltag ist die Paketsuche über die Webseite des Debian-Projekts (siehe [Debian-Paketsuche]) äußerst hilfreich und insbesondere sehr schnell. Abbildung 8.10 zeigt das Ergebnis der Recherche nach dem Paket *iftop* im Webbrowser *Iceweasel* an.

Neben dem Paketnamen beinhaltet jeder Suchtreffer die Distribution (siehe Abschnitt 2.9), gefolgt von der Veröffentlichung (hier genannt „Suite“) (siehe Abschnitt 2.10), der Paketkategorie (siehe Abschnitt 2.8) und den Debian-Architekturen (siehe Abschnitt 1.2), für die passende Pakete zur Verfügung stehen. Damit sehen Sie sofort, ob das Paket für ihre Veröffentlichung und Architektur existiert.

Klicken Sie einen der Links unterhalb des Suchfeldes an, schränken Sie das Suchergebnis auf die jeweilige Veröffentlichung oder Architektur weiter ein und erhalten daraufhin detailliertere Informationen zu dem spezifisch ausgewählten Paket. Neben der Paketbeschreibung sehen Sie die Debian Tags, die Paketabhängigkeiten und am rechten Rand weiterführende Informationen zum Paket. Dazu zählen ein Screenshot von screenshots.debian.net (sofern verfügbar), Fehlerberichte, die Liste der Änderungen („Changelog“), die Quellcodepakete, den Paketbetreuer („Maintainer“), die Projektwebseite und eine Liste ähnlicher Pakete.

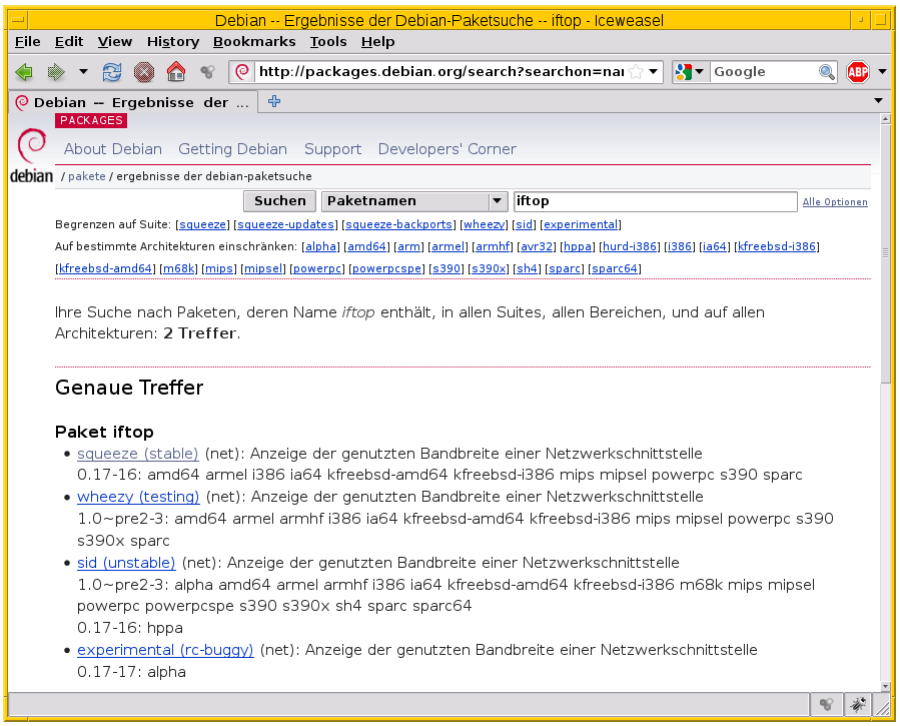

Abbildung 8.10: Ergebnis der Paketsuche nach *iftop* über http://packages.debian.org/

8.19.7 Suchmaschinen für Pakete

Bereits oben angesprochene Suche über die Webseite des Debian-Projekts beinhaltet nur Pakete, die in den offiziellen Repositories enthalten sind. Für andere, externe Pakete existieren hingegen spezielle Suchmaschinen und Verzeichnisdienste. Für `deb`-Pakete sind das bspw. apt-get.org [apt-get.org] und Rpmseek [rpmseek]. Letzeres kann sowohl `rpm`- als auch `deb`-Pakete erstöbern. Gefundene Pakete können Sie direkt von der angegebenen Quelle beziehen und installieren. Bitte beachten Sie aber, dass mit diesen Suchmaschinen gefundene Pakete oft nicht den Qualitätsansprüchen von Debian entsprechen, einer nicht-freien Lizenz unterliegen oder schlicht nicht auf Ihrem System installierbar sind, weil z.B. manche Abhängigkeiten nicht erfüllt werden.

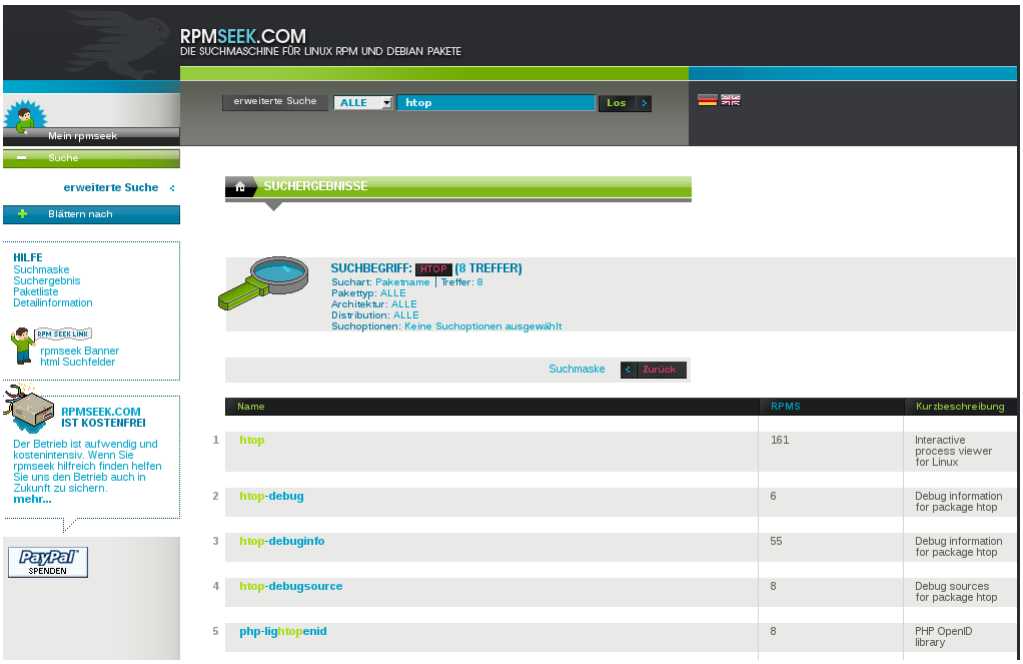

Abbildung 8.11: Ergebnis der Paketsuche nach *htop* über http://www.rpmseek.com/

Integration distributionsfremder deb-Pakete
Wie die Einbindung und Verifizierung von `deb`-Paketen aus den Paketquellen erfolgt, erklären wir Ihnen unter Paketquellen und Werkzeuge (Kapitel 3) genauer. Möchten sie auch `rpm`-Pakete einpflegen, sorgt der Abschnitt Fremdformate mit `alien` hinzufügen in Abschnitt 22.2 für Erleuchtung. Andere Paketformate betrachten wir im Buch nicht weiter.

8.20 Pakete über die Paketbeschreibung finden

Bleibt ihre Recherche über den Paketnamen ohne Erfolg, dehnen Sie ihre Suche am besten auf die gesamte Paketbeschreibung aus. Zur Recherche darin helfen Ihnen die darauf spezialisierten Programme `grep-available` und `grep-status` aus dem Paket *dctrl-tools* [Debian-Paket-dctrl-tools].

Ohne die Angabe weiterer Parameter durchsucht `grep-available` die gesamte Paketbeschreibung. Mit Hilfe der Option `-F` schränken Sie den Vorgang auf einen darüber ausgewählten Feldnamen ein. Nachfolgender Aufruf zeigt Ihnen von allen verfügbaren Paketen die Paketnamen an, bei denen in der Beschreibung und im Paketnamen die Zeichenfolge `xpdf` enthalten ist. Durch den Schalter `-i` erfolgt die Suche dabei unabhängig von der Groß- und Kleinschreibung. Das abschließende `sort`-Kommando sorgt darüberhinaus für eine Ausgabe in alphabetisch aufsteigender Abfolge.

Verfügbare Pakete anzeigen, bei denen in der Beschreibung und im Paketnamen die Zeichenfolge xpdf enthalten ist

```
$ grep-available -F Description -F Package -i xpdf | grep Package | sort
Package: flpsed
Package: libpoppler19
Package: libpoppler3
Package: libpoppler5
Package: libpoppler-cpp0
Package: libpoppler-glib3
Package: libpoppler-glib4
Package: libpoppler-glib8
Package: libpoppler-qt2
Package: libpoppler-qt4-3
Package: poppler-utils
Package: python-poppler
```

```
Package: xpdf
$
```

Obige Liste enthält alle Pakete – unabhängig davon, ob diese auf ihrem System installiert sind, oder nicht. Mit dem nachfolgenden Aufruf reduzieren Sie die Liste entsprechend. Dabei kommt der Schalter `-s` zum tragen, der den Paketstatus passend auswertet.

Lediglich die installierten Pakete anzeigen, bei denen in der Beschreibung und im Paketnamen die Zeichenfolge xpdf enthalten ist

```
$ grep-status -F Description -F Package -i -s Package xpdf | grep Package | sort
Package: flpsed
Package: libpoppler19
Package: libpoppler-cpp0
Package: libpoppler-glib8
Package: libpoppler-qt4-3
Package: poppler-utils
Package: python-poppler
Package: xpdf
$
```

(((grep-status, --invert-match)) Hilfreich ist auch der Schalter `-v` (Langversion `--invert-match`). Mit diesem verkehren Sie das Suchergebnis in das Gegenteil.

8.21 Paket nach Maintainer finden

Als *Debian Maintainer* versteht sich diejenige Einzelperson oder das Team, welche(s) eine Software betreut und für das entsprechende Debian-Paket verantwortlich zeichnet. Maintainer kümmern sich darum, dass das Paket gepflegt wird, d.h. Änderungen und Verbesserungen aus dem Upstream in das Paket unter Berücksichtigung der Debian-Spezifika einfließen, dieses dabei weiterhin den Debian-Richtlinien entspricht und in möglichst stabiler Form verfügbar bleibt [Debian-Wiki-Maintainer].

8.21.1 Welche Pakete betreut ein Debian-Maintainer

Diese Information liefern Ihnen mehrere Werkzeuge – `aptitude`, `grep-dctrl` sowie `ara` und Synaptic. Die Unterschiede liegen im Aufruf und den Parametern.

`aptitude` kennt den Schalter `~m` gefolgt von der Emailadresse des Maintainers. Damit finden Sie die *Binärpakete*, in denen dieser Maintainer als Verantwortlicher eingetragen ist. Für Axel Beckert und seine Emailadresse `abe@debian.org` lautet der komplette Aufruf `aptitude search '~m abe@debian.org'`. In der Schreibweise ist `aptitude` sehr tolerant – es gestattet den Aufruf mit und auch ohne Leerzeichen zwischen der Option und der Emailadresse.

Paketfilterung nach der Emailadresse des Maintainers mittels aptitude

```
$ aptitude search '~m abe@debian.org'
p   aha                              - Konvertiert ANSI-Farben nach HTML
p   amora-applet                     - use a bluetooth device as X remote control (
p   amora-cli                        - use a bluetooth device as X remote control (
p   autossh                          - SSH-Sitzungen und -Tunnel automatisch neu st
p   conkeror                         - Tastaturbedienbarer Webbrowser mit Emacs-ähn
p   conkeror-spawn-process-helper    - spawn external processes in Conkeror
p   dillo                            - Kleiner und schneller Webbrowser
p   dphys-config                     - Werkzeug zum Verteilen von Konfigurationsdat
p   dphys-swapfile                   - Automatisches Generieren und Nutzen einer Au
p   libapache2-mod-macro             - Create macros inside Apache config files
p   links                            - Textmodus-Webbrowser
i   links2                           - Webbrowser für den grafischen und den Textmo
...
$
```

Obige Ausgabe basiert auf dem Formatstring `-F '%c%a%M %p -%d'`. Die einzelnen Buchstaben stehen für den aktuellen Paketstatus (`%c`), die Aktion des Pakets (`%a`), automatische Installation (`%M`), den Paketnamen (`%p`) und die Paketbeschreibung (`%d`). Eine detaillierte Übersicht zu allen zulässigen Formatoptionen erklären wir Ihnen unter `aptitude` Format Strings in Abschnitt 10.4. Eine Alternative bietet zudem das Aptitude Handbuch [aptitude-dokumentation-paketdarstellung].

Möchten Sie in der Ausgabe stattdesen nur den Paketnamen, die Paketbeschreibung und den Paketmaintainer ausgeben, helfen Ihnen dabei die drei Optionen `%p`, `%d` und `%m` weiter. Letztgenanntes steht als Abkürzung für *maintainer*.

Paketsuche nach Paketnamen, Paketbeschreibung und Paketmaintainer

```
$ aptitude search -F '%p - %d - %m' '~mabe@debian.org'
aha             - Konvertiert ANSI-Farben nach HTML  - Axel Beckert <abe@debian.org>
amora-applet    - use a bluetooth device as X remote - Axel Beckert <abe@debian.org>
amora-cli       - use a bluetooth device as X remote - Axel Beckert <abe@debian.org>
autossh         - SSH-Sitzungen und -Tunnel automati - Axel Beckert <abe@debian.org>
...
$
```

Das Programm `grep-dctrl` aus dem Paket *dctrl-tools* sucht alle Pakete heraus, bei dem als Maintainer oder Uploader die angefragte Person hinterlegt ist und gibt diese zum gefundenen Paketnamen aus. Als weiteren Parameter benötigt es noch die lokal gespeicherte Paketliste, die es durchsuchen soll. Im nachfolgenden Beispiel ist das die Paketliste für den Paketmirror `ftp.de.debian.org` für die Distribution Debian 7 *Wheezy* sowie daraus der Bereich *main* und die Architektur *i386*.

Suche nach Maintainer und Uploader in der lokalen Paketliste (Ausschnitt)

```
$ grep-dctrl -F Maintainer,Uploaders abe@debian.org -s Package,Maintainer,Uploaders /var/ ↩
    lib/apt/lists/ftp.de.debian.org_debian_dists_wheezy_main_binary-i386_Packages
Package: aha
Maintainer: Axel Beckert <abe@debian.org>

Package: amora-applet
Maintainer: Axel Beckert <abe@debian.org>

Package: amora-cli
Maintainer: Axel Beckert <abe@debian.org>

Package: autossh
Maintainer: Axel Beckert <abe@debian.org>

...
$
```

Das Programm `ara` aus dem gleichnamigen Paket [Debian-Paket-ara] erlaubt eine sehr kompakte Schreibweise. Es sucht dabei in den Binär- und den Sourcepaketen, sofern diese vorhanden sind und gibt dabei trotzdem die Namen der dazugehörigen Binärpakete aus. Nachfolgender Aufruf zeigt die Recherche nach den Feldern *uploaders* und *maintainer* für den Benutzer mit der Emailadresse `abe@debian.org`. Bitte berücksichtigen Sie bei eigenen Experimenten die unterschiedlichen Trennzeichen zwischen den Komponenten und den abschließenden Schrägstrich nach der Emailadresse.

Einfache Recherche nach Paketen mit Hilfe von ara

```
$ ara uploaders,maintainer:/abe@debian.org/
aha
amora-applet
amora-cli
amora-server
autocutsel
autossh
conkeror
conkeror-spawn-process-helper
...
$
```

Besonders schön sehen Sie das bei der nachfolgenden Abfrage in Abbildung 8.12. Die verwendeten Schalter `-progress` und `-table` zählen nicht nur die überprüften Pakete und zeigen die gerade untersuchte Paketquelle an, sondern zeichnen auch noch eine hübsche Tabelle ringsrum. Zu jedem gefundenen Source-Paket werden in der jeweiligen Spalte die dazugehörigen Binärpakete aufgelistet.

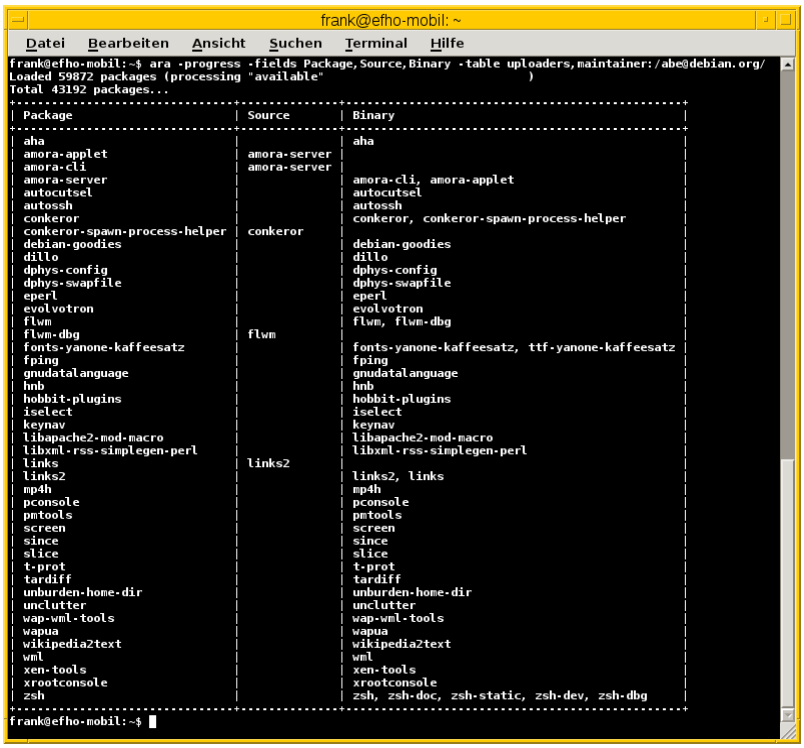

Abbildung 8.12: Recherche nach Paketen mittels `ara`

Das graphische Programm Synaptic (Abschnitt 6.4.1) handhabt das ganze etwas anders und bietet Ihnen einen passenden Menüeintrag an. Unter dem Eintrag Bearbeiten → Suchen bzw. mit der Tastenkombination Ctrl-F erreichen Sie den Suchdialog. Im Auswahlfeld selektieren Sie den Eintrag Betreuer und tragen im Eingabefeld dessen Namen ein. Daraufhin liefert Ihnen Synaptic ein Ergebnis wie in Abbildung 8.13. In der linken Spalte der Paketauswahl erscheint zudem ein zusätzlicher Eintrag mit dem Namen des Paketmaintainers.

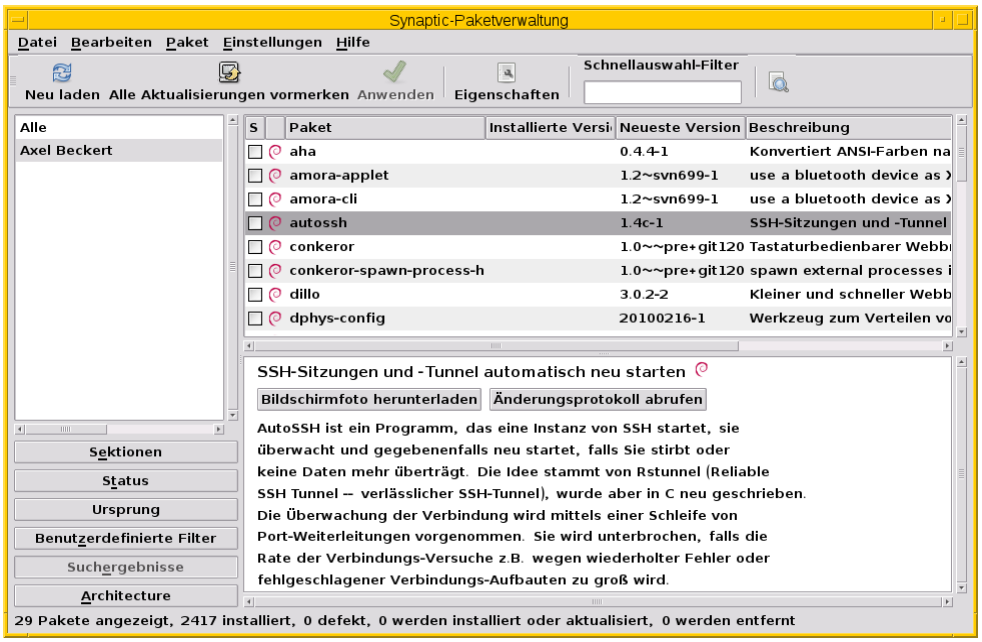

Abbildung 8.13: Ergebnis der Suche nach dem Paketmaintainer in Synaptic

8.21.2 Rückrichtung: Wer betreut ein bestimmtes Paket

Interessant ist natürlich auch die Rückrichtung: das Ausgeben aller Maintainer und Co-Maintainer zu einer Liste von Source- und Binärpaketen. Das gelingt Ihnen mit dem Kommando `dd-list` aus dem Paket *devscripts* [Debian-Paket-devscripts]. Als Parameter geben Sie die Namen der Pakete an, die Sie interessieren. Leider werden in der Ausgabe die Co-Maintainer irreführend als Uploader mit einem großen `U` benannt.

Ausgabe der Maintainer und Co-Maintainer mittels dd-list

```
$ dd-list screen xymon fping ack-grep
Anibal Monsalve Salazar <anibal@debian.org>
   fping

Axel Beckert <abe@debian.org>
   ack-grep (U)
   fping (U)
   screen
   xymon (U)

Christoph Berg <myon@debian.org>
   xymon

Debian Perl Group <pkg-perl-maintainers@lists.alioth.debian.org>
   ack-grep
Jan Christoph Nordholz <hesso@pool.math.tu-berlin.de>
   screen (U)

Ryan Niebur <ryan@debian.org>
   ack-grep (U)
$
```

Viele Entwickler mögen dieses Kommando sehr. Sie verwenden es, um Listen von einem bestimmten Problem oder einer Migration betroffenen Pakete zu erhalten. Darin suchen sie nach ihrem eigenen Namen und wenn dieser nicht mehr darin auftaucht, haben sie keine Arbeit mehr damit ;-)

8.22 Paket zu Datei finden

Häufig ist Ihnen nur der Dateiname bekannt, aber nicht das Paket, aus dem diese Datei stammt. Das wird insofern spannend, wenn das Paket anders als die gesuchte Datei heißt. Es gibt derzeit fünf Möglichkeiten, diese Zuordnung zu ermitteln – einerseits über `dpkg`, `dpkg-query` oder `dlocate` [Debian-Paket-dlocate] mit deren Option `-S Muster`, andererseits mittels `apt-file` und dessen Option `search Muster` und fünftens über die Webseite des Debian-Projekts. `aptitude` verfügt nicht über einen solchen Schalter zur Suche.

Die vier Kommandos finden alle Pfade, in denen das angegebene „Muster“ vorkommt. Der Unterschied zwischen den vier Programmen besteht darin, dass `dpkg`, `dpkg-query` und `dlocate` nur in bereits installierten Paketen suchen, `apt-file` hingegen hingegen in allen verfügbaren Paketen, d.h. unabhängig davon, ob diese bereits auf ihrem System installiert sind oder nicht. Verfügbare Pakete bezeichnet die Menge von Paketen, die APT über einen Eintrag in der Liste der Paketquellen in der Datei `/etc/apt/sources.list` und aus der dazugehörigen Paketliste entnehmen kann (siehe dazu Abschnitt 3.1). Pakete, die sich hingegen in Paketquellen befinden, die nicht in obiger Liste referenziert sind, kann `apt-file` nicht untersuchen.

Bei der Suche über die Webseite bildet zunächst der gesamte Paketbestand aller Veröffentlichungen die Grundlage. Sie können das jederzeit entsprechend über die Auswahlfelder zur Veröffentlichung oder Architektur einschränken.

8.22.1 Suche in bereits installierten Paketen

Dafür genügen der Aufruf `dpkg -S Muster`, `dpkg-query -S Muster` oder die flinke Abkürzung `dlocate Muster`. Zur Suche ist bei `dlocate` der Schalter `-S` optional, `dpkg` und `dpkg-query`kennen hingegen dafür die Langf orm `--search`. Das Textfragment oder Muster beschreibt, wonach die Programme in der Dateiliste der installierten Pakete suchen sollen. Beginnt das Textfragment mit einem `/`, wird die Zeichenkette als absoluter Pfad interpretiert. Nachfolgendes Beispiel illustriert den Aufruf nach dem Muster `aptsh` mittels `dpkg` in allen installierten Paketen.

Suche nach dem Muster aptsh mittels dpkg

```
$ dpkg -S aptsh
aptsh: /usr/lib/aptsh/aptsh_ls
aptsh: /usr/share/doc/aptsh/changelog.gz
aptsh: /usr/share/man/man1/aptsh.1.gz
aptsh: /usr/share/doc/aptsh/copyright
aptsh: /usr/share/doc/aptsh
aptsh: /etc/aptsh.conf
aptsh: /usr/lib/aptsh/aptsh_printer
aptsh: /usr/bin/aptsh
aptsh: /usr/share/doc/aptsh/HOWTO.gz
aptsh: /usr/lib/aptsh/aptsh_rls
aptsh: /usr/lib/aptsh
$
```

`dlocate` liefert im Allgemeinen ein identisches Ergebnis zu `dpkg` bzw. `dpkg-query`. Da `dlocate` zur Suche auf `grep` zurückgreift, hat es mitunter eine höhere Trefferrate. Nachfolgendes Beispiel zeigt die Suche nach dem absoluten Pfad `/xara-gtk` – sowohl für `dpkg`, als auch zu `dlocate` im Vergleich.

Suche nach der Pfadangabe /xara-gtk

```
$ dpkg -S /xara-gtk
dpkg-query: Kein Pfad gefunden, der auf Muster /xara-gtk passt
$ dlocate -S /xara-gtk
xara-gtk: /etc/xara-gtkrc-2.0
xara-gtk: /usr/share/menu/xara-gtk
xara-gtk: /usr/share/doc/xara-gtk
xara-gtk: /usr/share/doc/xara-gtk/copyright
xara-gtk: /usr/share/doc/xara-gtk/changelog.gz
$
```

8.22.2 Suche in noch nicht installierten Paketen

Dafür gibt es `apt-file`. Grundlage seiner Aktivitäten ist die Liste der Paketquellen in `/etc/apt/sources.list/` und die Liste der Dateien in jedem der Pakete, die von dort bezogen werden. Letztere befindet sich im Verzeichnis `/var/cache/apt/apt-file/`.

Liste der Paketquellen anzeigen

```
$ ls /var/cache/apt/apt-file
ftp.de.debian.org_debian_dists_wheezy_contrib_Contents-i386.gz
ftp.de.debian.org_debian_dists_wheezy_main_Contents-i386.gz
ftp.de.debian.org_debian_dists_wheezy_non-free_Contents-i386.gz
httpredir.debian.org_debian_dists_wheezy_contrib_Contents-i386.gz
httpredir.debian.org_debian_dists_wheezy_main_Contents-i386.gz
httpredir.debian.org_debian_dists_wheezy_non-free_Contents-i386.gz
$
```

Sollte diese Liste noch nicht vorhanden sein, erhalten Sie diese mit dem Aufruf `apt-file update`. Danach bezieht `apt-file` ein durchaus größeres Archiv (20 MB) von den angegebenen Paketmirrors und legt diese Dateien im Verzeichnis `/var/cache/apt/apt-file/` ab.

Im nachfolgenden Beispiel zu `apt-file update` wird der Paketmirror dynamisch ausgewählt. Hintergrundinformationen dazu finden Sie in Abschnitt 3.3 und Abschnitt 3.6.2:

Suchdatenbank von APT aktualisieren

```
# apt-file update
Downloading complete file http://httpredir.debian.org/debian/dists/wheezy/main/Contents- ↩
    i386.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 20.6M  100 20.6M    0     0  3917k      0  0:00:05  0:00:05 --:--:-- 4111k
Downloading complete file http://httpredir.debian.org/debian/dists/wheezy/contrib/Contents- ↩
    i386.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 73552  100 73552    0     0   302k      0 --:--:-- --:--:-- --:--:--  302k
Downloading complete file http://httpredir.debian.org/debian/dists/wheezy/non-free/Contents ↩
    -i386.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  723k  100  723k    0     0  1428k      0 --:--:-- --:--:-- --:--:-- 1428k
Ignoring source without Contents File:
  http://security.debian.org/dists/wheezy/updates/main/Contents-i386.gz
Ignoring source without Contents File:
  http://security.debian.org/dists/wheezy/updates/contrib/Contents-i386.gz
Ignoring source without Contents File:
  http://security.debian.org/dists/wheezy/updates/non-free/Contents-i386.gz
#
```

`apt-file` verfügt über eine ganze Reihe von Unterkommandos und Schaltern, von denen wir Ihnen die wichtigsten vorstellen. Weitere Schalter entnehmen Sie bitte der Manpage zum Programm.

`search Muster`
: Suche danach, in welchem Paket die als `Muster` angegebene Datei enthalten ist. Ergebnis ist eine Liste aller Pakete, die das angegebene `Muster` enthalten. `apt-file` sucht dabei nur nach Dateinamen, nicht jedoch nach Verzeichnisnamen.

`find Muster`
: Alias für den Schalter `search`.

`list Muster`
gibt den Paketinhalt aus, auf den das Muster passt. Diese Aktion ist sehr ähnlich zum Aufruf `dpkg -L`, nur das hier die Pakete noch nicht installiert sein müssen.

`show Muster`
Alias für den Schalter `list`.

`-a` (Langform `--architecture`)
Einschränkung der Suche auf die angegebene Architektur (siehe Abschnitt 1.2).

`-i` (Langform `--ignore-case`)
Suche unabhängig von Groß- und Kleinschreibung des Musters.

`-l` (Langform `--package-only`)
Das Ergebnis ist nur der Paketname, auf den das Muster passt. Dateinamen werden nicht berücksichtigt.

`-x` (Langform `--regexp`)
interpretiert das Muster als Regulären Ausdruck, so wie ihn Perl versteht (PCRE). Ohne diesen Schalter wird das Muster als schlichte Zeichenkette aufgefasst.

`-v` (Langform `--verbose`)
verbose, d.h. die Ausgabe wird deutlich ausführlicher.

Etwas nachteilig an `apt-file` ist, dass es alle Paketquellen durchsucht und Ihnen dabei nicht anzeigt, in welcher davon es den Treffer gefunden hat. Das führt zu Verwirrung, bspw. wenn in der Liste der Paketquellen die Veröffentlichungen *stable* und *stable-backports* eingetragen sind. `apt-file` verfügt bislang nicht über einen Schalter, um die Ausgabe dementsprechend zu beeinflussen.

Aktuelle Strukturdatenbank
Um vernünftig mit `apt-file` arbeiten zu können, empfehlen wir Ihnen, zuerst mit `apt-file update` die bestehende Dateiliste zu aktualisieren und danach darin zu stöbern. Damit nutzen Sie eine aktuelle Datenbasis.

Das nachfolgende Beispiel zeigt die Suche der Zeichenkette `aptsh`. Zusätzlich kommt der Schalter `-v` (Langform `--verbose`) zum Einsatz, um eine ausführlichere Ausgabe zu erhalten.

Suche über die Strukturdatenbank mittels apt-file

```
# apt-file -v show aptsh
D: Using cache directory /var/cache/apt/apt-file
D: reading sources file /etc/apt/sources.list
D: got 'deb http://httpredir.debian.org/debian/ wheezy main contrib non-free'
D: kept 'deb http://httpredir.debian.org/debian/ wheezy main contrib non-free'
D: got 'deb http://security.debian.org/ wheezy/updates main contrib non-free'
D: kept 'deb http://security.debian.org/ wheezy/updates main contrib non-free'
D: regexp: ^\s*(.*?)\s+(\S*/\S*aptsh\S*)\s*$
D: Search in \/var\/cache\/apt\/apt\-file\/httpredir\.debian\. ↩
    org_debian_dists_wheezy_main_Contents\-i386\.gz using zfgrep  -- aptsh
.........
D: Search in \/var\/cache\/apt\/apt\-file\/httpredir\.debian\. ↩
    org_debian_dists_wheezy_contrib_Contents\-i386\.gz using zfgrep  -- aptsh

D: Search in \/var\/cache\/apt\/apt\-file\/httpredir\.debian\.org_debian_dists_wheezy_non\- ↩
    free_Contents\-i386\.gz using zfgrep  -- aptsh

aptsh: /etc/aptsh.conf
aptsh: /usr/bin/aptsh
aptsh: /usr/lib/aptsh/aptsh_ls
aptsh: /usr/lib/aptsh/aptsh_printer
aptsh: /usr/lib/aptsh/aptsh_rls
aptsh: /usr/share/doc/aptsh/HOWTO.gz
```

```
aptsh: /usr/share/doc/aptsh/changelog.gz
aptsh: /usr/share/doc/aptsh/copyright
aptsh: /usr/share/man/man1/aptsh.1.gz
#
```

8.22.3 Suche über die Webseite des Debian-Projekts

Die Webseite bietet ebenfalls eine Suche anhand einer Zeichenfolge an (siehe Abbildung 8.14). Über verschiedene Auswahlfelder grenzen Sie ein, ob die Zeichenfolge auf feste Verzeichnisse passen soll, die mit einem Suchwort enden oder Pakete mit Dateien beinhalten soll, die so benannt sind oder deren Namen das Suchwort enthalten. Desweiteren filtern Sie die Suchergebnisse nach der gewünschten Veröffentlichung und Architektur (siehe dazu Abschnitt 2.10 und Abschnitt 1.2).

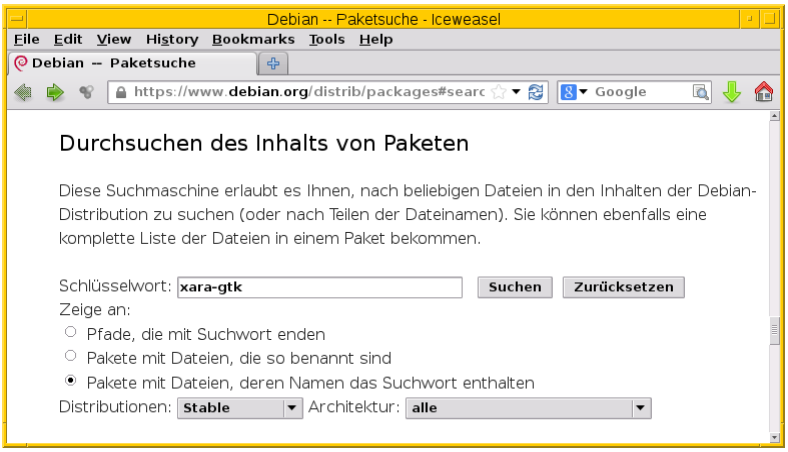

Abbildung 8.14: Suche nach `xara-gtk` über die Webseite

Die Abbildung 8.15 zeigt das Suchergebnis für die Veröffentlichung *Wheezy*, welches hier recht übersichtlich ausfällt. Beide Treffer zeigen das Paket *xara-gtk* samt der dazu gefundenen Dateien mit dem Suchmuster. Klicken Sie auf einen der Links zwischen dem Suchfeld und dem Suchergebnis, schränken Sie die Suche anhand der gewählten Veröffentlichung bzw. Architektur weiter ein.

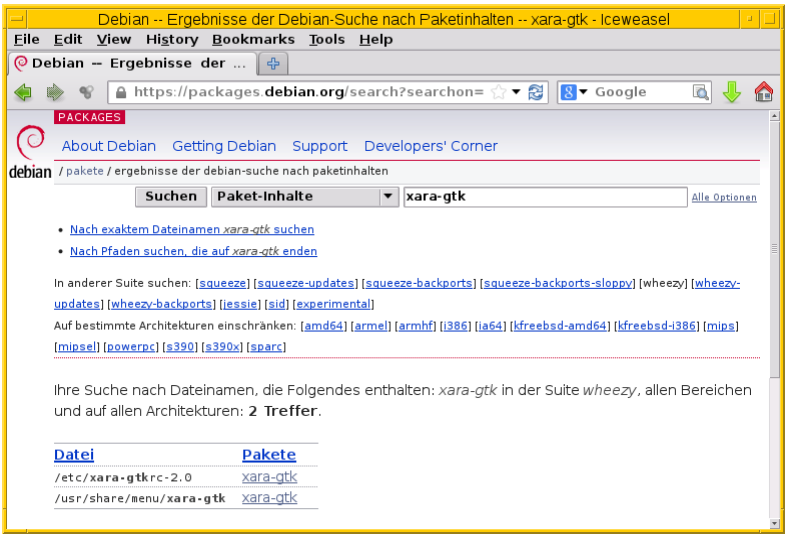

Abbildung 8.15: Suche nach dem Paket *xara-gtk* über die Webseite des Debian-Projekts (Suchergebnis)

8.23 Paketinhalte anzeigen (`apt-file`)

In einem Paket sind stets mehrere Dateien zusammengefasst. Mit den sechs Werkzeugen `dpkg`, `dpkg-deb`, `dpkg-query`, `dlocate`, `apt-file` und `dglob` zeigen Sie den Inhalt eines Pakets an. Dabei sind `dpkg-deb` und `dpkg-query` Hilfsprogramme von `dpkg` und verstehen die gleichen Schalter.

Es sind mehrere Fälle zu unterscheiden, die jeweils unterschiedliche Aufrufe nachsichziehen:

das Paket ist bereits installiert
: `dpkg -L` *Paketname*, `dpkg-query -L` *Paketname*, `dlocate -ls` *Paketname* sowie mittels `dglob -f` *Paketname*. Der Parameter *Paketname* bezeichnet lediglich den Namen des Pakets (siehe Abschnitt 2.11) ohne Angabe der Versionsnummer.

das Paket ist noch nicht installiert
: `dpkg -c` *deb-Datei* oder `dpkg-deb -c` *deb-Datei*. Der Parameter *deb-Datei* ist ein Paketarchiv in Form einer lokal vorliegenden Datei. Befindet sich die Datei nicht im aktuellen Verzeichnis, von dem aus Sie das Kommando aufrufen, ergänzen Sie im Aufruf den dazugehörigen Verzeichnispfad, in dem das Paketarchiv liegt.

das Paket muss nicht installiert sein, kann aber
: `apt-file show` *Paketname*, `apt-file list` *Paketname* und `dglob -af` *Paketname*. Der Parameter *Paketname* bezeichnet hier lediglich den Namen eines Pakets (siehe Abschnitt 2.11) ohne Angabe der Versionsnummer.

8.23.1 `dpkg -L` *Paketname*

Die Langform des Schalters ist `--listfiles`. Beide Schalter versteht ebenso das Hilfsprogramm `dpkg-query` und erzeugt die gleiche Ausgabe. Damit listen Sie den Paketinhalt mit allen Pfaden auf. Jede Verzeichnisebene ist separat aufgeführt. Das nachfolgende Beispiel verdeutlicht das am Paket *xara-gtk*.

Auflistung des Paketinhalts mit allen Pfaden via dpkg

```
$ dpkg -L xara-gtk
/.
/etc
/etc/xara.config
/etc/xara-gtkrc-2.0
/usr
/usr/bin
/usr/bin/xara
/usr/share
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/xara.1.gz
/usr/share/menu
/usr/share/menu/xara-gtk
/usr/share/doc
/usr/share/doc/xara-gtk
/usr/share/doc/xara-gtk/copyright
/usr/share/doc/xara-gtk/changelog.gz
$
```

8.23.2 `dlocate -L` *Paketname*

Eine identische Ausgabe zum vorherigen `dpkg`-Aufruf ermöglicht Ihnen das Programm `dlocate` [Debian-Paket-dlocate] mit dem Schalter `-L`. Beachten Sie hierbei jedoch, dass `dlocate` die Angabe des Paketnamens als regulären Ausdruck interpretiert.

8.23.3 `dlocate --ls` *Paketname*

Nutzen Sie statt `-L` hingegen den Schalter `-ls`, wird die Ausgabe sehr ausführlich. Es entspricht dem Aufruf des UNIX-Kommandos `ls -ldF` bezogen auf alle Dateien, die in dem Paket enthalten sind.

Auflistung des Paketinhalts in ausführlicherer Form via dlocate

```
$ dlocate -ls xara-gtk
drwxr-xr-x   24 root root    4096 Mär 13 09:44 /./
drwxr-xr-x  182 root root   12288 Jun 22 23:21 /etc/
-rw-r--r--    1 root root     658 Jun 11  2010 /etc/xara.config
-rw-r--r--    1 root root     136 Jun 11  2010 /etc/xara-gtkrc-2.0
drwxr-xr-x   11 root root    4096 Jan  8 00:44 /usr/
drwxr-xr-x    2 root root  110592 Jun 13 16:34 /usr/bin/
-rwxr-xr-x    1 root root 1828064 Mai 21  2012 /usr/bin/xara*
drwxr-xr-x  396 root root   12288 Jun 13 16:34 /usr/share/
drwxr-xr-x 2292 root root   77824 Jun 13 16:34 /usr/share/doc/
drwxr-xr-x    2 root root    4096 Jan  8 01:55 /usr/share/doc/xara-gtk/
-rw-r--r--    1 root root    5488 Mai 20  2012 /usr/share/doc/xara-gtk/changelog.gz
-rw-r--r--    1 root root    1281 Apr 18  2011 /usr/share/doc/xara-gtk/copyright
drwxr-xr-x   50 root root    4096 Jan  8 01:56 /usr/share/man/
drwxr-xr-x    2 root root  131072 Jun 13 16:34 /usr/share/man/man1/
-rw-r--r--    1 root root    6121 Mai 21  2012 /usr/share/man/man1/xara.1.gz
drwxr-xr-x    2 root root    4096 Jun 13 16:34 /usr/share/menu/
-rw-r--r--    1 root root     160 Apr 18  2011 /usr/share/menu/xara-gtk
$
```

8.23.4 `dpkg -c` *deb-Datei*

Sie verwenden den Schalter `-c`, um sich den Inhalt eines `deb`-Pakets anzeigen zu lassen (Langform `--contents`). Dieses Paket wird `dpkg` als Parameter übergeben und kann sowohl eine Datei in einem lokalen Verzeichnis bezeichnen, als auch den Namen eines Archivs. Im Gegensatz zu `dpkg -L` muss das Paket nicht auf ihrem System installiert sein. Intern übergibt `dpkg` die Ausführung an `dpkg-deb`, welches Sie auch separat aufrufen können.

Auflistung des Paketinhalts mit allen Informationen via dpkg

```
$ dpkg -c /var/cache/apt/archives/xara-gtk_1.0.31_i386.deb
drwxr-xr-x root/root         0 2012-05-21 01:04 ./
drwxr-xr-x root/root         0 2012-05-21 01:04 ./etc/
-rw-r--r-- root/root       658 2011-04-18 19:29 ./etc/xara.config
-rw-r--r-- root/root       136 2011-04-18 19:29 ./etc/xara-gtkrc-2.0
drwxr-xr-x root/root         0 2012-05-21 01:04 ./usr/
drwxr-xr-x root/root         0 2012-05-21 01:04 ./usr/bin/
-rwxr-xr-x root/root   1828064 2012-05-21 01:04 ./usr/bin/xara
drwxr-xr-x root/root         0 2012-05-21 01:04 ./usr/share/
drwxr-xr-x root/root         0 2012-05-21 01:04 ./usr/share/man/
drwxr-xr-x root/root         0 2012-05-21 01:04 ./usr/share/man/man1/
-rw-r--r-- root/root      6121 2012-05-21 01:04 ./usr/share/man/man1/xara.1.gz
drwxr-xr-x root/root         0 2012-05-21 01:04 ./usr/share/menu/
-rw-r--r-- root/root       160 2011-04-18 19:28 ./usr/share/menu/xara-gtk
drwxr-xr-x root/root         0 2012-05-21 01:04 ./usr/share/doc/
drwxr-xr-x root/root         0 2012-05-21 01:04 ./usr/share/doc/xara-gtk/
-rw-r--r-- root/root      1281 2011-04-18 19:28 ./usr/share/doc/xara-gtk/copyright
-rw-r--r-- root/root      5488 2012-05-20 23:18 ./usr/share/doc/xara-gtk/changelog.gz
$
```

8.23.5 `apt-file show` *Paketname* und `apt-file list` *Paketname*

Die beiden Optionen `show` und `list` des Werkzeugs `apt-file` sind synonym zueinander und liefern den gleichen Inhalt wie `dpkg -L`. Dabei ist die Darstellung von `apt-file` jedoch deutlich kompakter.

Paketinhalt in kompakter Form mittels apt-file

```
$ apt-file show xara-gtk
xara-gtk: /etc/xara-gtkrc-2.0
xara-gtk: /etc/xara.config
xara-gtk: /usr/bin/xara
xara-gtk: /usr/share/doc/xara-gtk/changelog.gz
xara-gtk: /usr/share/doc/xara-gtk/copyright
xara-gtk: /usr/share/man/man1/xara.1.gz
xara-gtk: /usr/share/menu/xara-gtk
$
```

8.23.6 Einsatz von `dglob`

Analog zu `apt-file` arbeitet das Werkzeug `dglob` aus dem Paket *debian-goodies* [Debian-Paket-debian-goodies]. Die Ausgabe ist ähnlich kompakt wie von `apt-file`. Der Schalter `-f` dient dabei zur Ausgabe der Dateien im angefragten Paket, was wir nachfolgend erneut anhand des Pakets *xara-gtk* illustrieren.

Ergebnis der Recherche zum Paket xara-gtk

```
$ dglob -f xara-gtk
/etc/xara.config
/etc/xara-gtkrc-2.0
/usr/bin/xara
/usr/share/man/man1/xara.1.gz
/usr/share/menu/xara-gtk
/usr/share/doc/xara-gtk/copyright
/usr/share/doc/xara-gtk/changelog.gz
$
```

Das Kommando `dglob` agiert üblicherweise nur auf den bereits installierten Paketen. Mit dem Schalter `-a` weiten Sie Ihre Recherche auf alle verfügbaren Pakete aus — auch auf diejenigen, die noch nicht installiert sind. Für diesen Schritt setzt `dglob` auf das Programm `grep-aptavail` aus dem Paket *dctrl-tools* [Debian-Paket-dctrl-tools] auf. Nähere Informationen zu *dctrl-tools* erfahren Sie unter Kapitel 13.

8.24 Nach Muster in einem Paket suchen

Zur Lösung dieser Aufgabe steht Ihnen das Programm `dgrep` mit seinen drei Varianten `degrep`, `dfgrep` und `dzgrep` aus dem Paket *debian-goodies* [Debian-Paket-debian-goodies] zur Verfügung. Dieses Shellskript kombiniert das Programm `dpkg` mit dem Suchwerkzeug `grep` und dessen Kollegen `egrep`, `fgrep` und `zgrep` miteinander. Je nach Bedarf suchen Sie darüber entweder in den Dateien aller bereits installierten Pakete oder lediglich in einer Auswahl davon. Nutzen Sie `dzgrep`, werden auch komprimierte Dateien in die Recherche mit einbezogen. Die Auflösung, welche Datei zu einem Paket gehört, erfolgt über das Programm `dglob` aus dem gleichen Paket.

Aufgrund der Verknüpfung der Programme können Sie zur Recherche nach dem gesuchten Muster auch die meisten der Optionen, die Sie von den `grep`-Varianten her kennen, einsetzen. Das schließt bspw. reguläre Ausdrücke und die farbige Hervorhebung der Suchtreffer in der Ausgabe mit ein. Ausgenommen sind jedoch Verzeichnisse und das Verfolgen von symbolischen Links.

In der nachfolgenden Ausgabe sehen Sie einen Ausschnitt des Rechercheergebnisses nach dem Muster `regular` im Paket *bash-doc*. Dabei beinhaltet die linke Spalte die Datei, in welcher das Muster auftrat, und in der rechten Spalte das Muster samt Kontext drumherum.

Suche nach dem Vorkommen des Musters regular im Paket bash-doc

```
$ dgrep --color regular bash-doc
/usr/share/doc/bash/examples/scripts.v2/where:    # Find all pattern matches that are  ↩
    executable regular files.
/usr/share/doc/bash/examples/complete/bash_completion:                # so we can set  ↩
    them before handing off to regular
```

```
/usr/share/doc/bash/examples/scripts/bcsh.sh:#  A cshell-style "setenv" command is turned  ↩
    into a regular "set" command.
...
$
```

Benötigen Sie hingegen nur eine kurze Liste mit den Dateinamen, hilft Ihnen die `grep`-Option `-l` (Langfassung `--files-with-matches`) weiter. Für zusätzliche Optionen werfen Sie bitte einen Blick in die Manpage zu `grep`. Nachfolgend sehen Sie einen Ausschnitt des Suchergebnisses nach dem Muster `regular` über alle installierten Pakete ohne Berücksichtigung der Groß- und Kleinschreibung (Option `-i` bzw. `--ignore-case` in der Langfassung).

Suche nach dem Vorkommen des Musters regular in allen installierten Paketen (Kurzfassung)

```
$ dgrep -l -i regular bash-doc
/usr/lib/perl5/XML/LibXML/Error.pm
/usr/lib/perl5/XML/LibXML/XPathContext.pod
/usr/lib/perl5/XML/LibXML/Text.pod
/usr/lib/perl5/XML/LibXML/RegExp.pod
/usr/share/doc/module-assistant/index.html
/usr/share/doc/libfftw3-3/README.Debian
/usr/share/perl5/Text/WrapI18N.pm
/usr/share/doc/chromium/README.source
/usr/share/doc/bash/examples/scripts.v2/where
/usr/share/doc/bash/examples/complete/bash_completion
/usr/share/doc/bash/examples/scripts/bcsh.sh
...
$
```

8.25 Ausführbare Dateien anzeigen

Die ausführbaren Dateien Ihres Linuxsystems befinden sich üblicherweise im Verzeichnis `/usr/bin` bzw. `/usr/sbin`. Um herauszufinden, welche ausführbaren Dateien sich in einem Paket befinden, können Sie einerseits das Paket durchforsten (siehe „Paketinhalte anzeigen“ in Abschnitt 8.23) oder andererseits das Kommando `dlocate` benutzen. Über die Option `-lsbin` und den Paketnamen gibt es Ihnen ausführlich Auskunft. Die nachfolgende Ausgabe zeigt die ausführbaren Dateien zum Paket *aptitude* an:

Ausführbare Dateien zum Paket aptitude anzeigen

```
$ dlocate -lsbin aptitude
/etc/cron.daily/aptitude
/usr/bin/aptitude-curses
/usr/share/bug/aptitude
$
```

Eine weitere Möglichkeit stellt das UNIX-Kommandos `whereis` aus dem essentiellen Paket *util-linux* [Debian-Paket-util-linux] dar. Mit der Option `-b Programmname` sucht `whereis` nach den passenden Binärdateien zum genannten Paketnamen.

Binärdateien mit dem Namen aptitude mittels whereis anzeigen

```
$ whereis -b aptitude
aptitude: /usr/bin/aptitude /usr/bin/X11/aptitude /usr/share/aptitude
$
```

8.26 Manpages anzeigen

Für die meisten UNIX/Linux-Werkzeuge bestehen Informations- und Hilfeseiten, auch genannt *Info und Man(ual) Pages*. Um in Erfahrung zu bringen, ob diese überhaupt vorhanden und installiert sind, bieten sich zunächt die beiden UNIX-Kommandos `apr`

`opos` und `whereis` (Paket *util-linux* [Debian-Paket-util-linux]) an. Ebenso hilft Ihnen das bereits mehrfach genutzte Werkzeug `dlocate` [Debian-Paket-dlocate] weiter.

Manpages aus `deb`-Paketen, die noch nicht auf ihrem System installiert sind, aber als lokale Datei vorliegen, zeigen Sie mit Hilfe von `debman` und `debmany` aus dem Paket *debian-goodies* [Debian-Paket-debian-goodies] an. Liegt das Paket nicht lokal vor, hilft Ihnen die Manpages-Sammlung des Debian-Projekts weiter [Debian-Manpages].

8.26.1 Manpages erstöbern

Mittels `apropos` *Paketname* sehen Sie, ob zu dem von Ihnen angefragten Programm lokal Dokumentation verfügbar ist. Für das Stichwort `aptitude` sieht das bspw. wie folgt aus:

Verfügbare Manpages für das Paket aptitude mittels aptitude lokalisieren

```
$ apropos aptitude
aptitude (8)         - Benutzerschnittstelle für den Paketmanager
aptitude-curses (8)  - Benutzerschnittstelle für den Paketmanager
aptitude-create-state-bundle (1) - bundle the current aptitude state
aptitude-run-state-bundle (1) - unpack an aptitude state bundle and invoke aptitude on it
$
```

Eine ähnliche Hilfe leistet auch das Kommando `dlocate` mit dem Schalter `-man` gefolgt vom Paketnamen. Das Ergebnis des Aufrufs sieht für das Programm `aptitude` sieht wie folgt aus:

Verfügbare Manpages für das Paket aptitude mittels dlocate aufspüren

```
$ dlocate -man aptitude
8 aptitude-curses
$
```

Nun können Sie die Manpage mittels `man aptitude` bzw. `man aptitude-curses` aufrufen.

Benötigen Sie zusätzlich den exakten Pfad zur Datei, in der die Manpage liegt, nutzen Sie stattdessen entweder `whereis` mit dem Schalter `-m` oder `dlocate` mit dem Schalter `-lsman`.

Verfügbare Manpages für das Paket aptitude mittels whereis lokalisieren

```
$ whereis -m aptitude
aptitude: /usr/share/man/man8/aptitude.8.gz
$
```

Bei letzterem erfahren Sie bspw. aus dem nachfolgenden Aufruf, dass die Manpage für mehrere Sprachen wie bspw. Deutsch (de), Spanisch (es) und Polnisch (pl) im Verzeichnis `/usr/share/man` bereitsteht.

Verfügbare Manpages für das Paket aptitude mit vollständigem Pfad

```
$ dlocate -lsman aptitude
/usr/share/man/cs/man8/aptitude-curses.8.gz
/usr/share/man/es/man8/aptitude-curses.8.gz
/usr/share/man/ja/man8/aptitude-curses.8.gz
/usr/share/man/fr/man8/aptitude-curses.8.gz
/usr/share/man/gl/man8/aptitude-curses.8.gz
/usr/share/man/fi/man8/aptitude-curses.8.gz
/usr/share/man/man8/aptitude-curses.8.gz
/usr/share/man/it/man8/aptitude-curses.8.gz
/usr/share/man/pl/man8/aptitude-curses.8.gz
/usr/share/man/de/man8/aptitude-curses.8.gz
$
```

8.26.2 Manpages aus noch nicht installierten Paketen anzeigen

Für diesen speziellen Zweck existieren `debman` und `debmany`. Beide Werkzeuge gehören zum Umfang des Pakets *debian-goodies* [Debian-Paket-debian-goodies] und bieten Ihnen die Möglichkeit, Manpages von Paketen anzuzeigen, die noch nicht installiert sind, aber bspw. bereits als lokale Datei vorliegen. Dazu benötigt `debman` den Schalter `-f`, den Dateinamen und anschließend den Namen der Manpage. Der nachfolgende Aufruf am Beispiel von *htop* offnet den bei Ihnen voreingestellten Betrachter für Manpages — ganz so, als ob das Paket tatsächlich installiert wäre.

Aufruf von debman zum Paket htop

```
$ debman -f htop_1.0.3-1_amd64.deb htop
...
$
```

Liegt das Paket hingegen noch nicht lokal vor, gibt es auch keinen Grund zur Verzweiflung. Sowohl `debman` als auch `debmany` können dieses vom Paketmirror beziehen und die Darstellung der Manpage daraus veranlassen. `debman` versteht dazu den Schalter `-p`.

Sind Sie mit dem Webbrowser unterwegs und bevorzugen diese Darstellung, greifen Sie darüber auf die Manpages-Sammlung des Debian-Projekts zurück [Debian-Manpages]. Abbildung 8.16 zeigt das Ergebnis der Recherche nach `htop`.

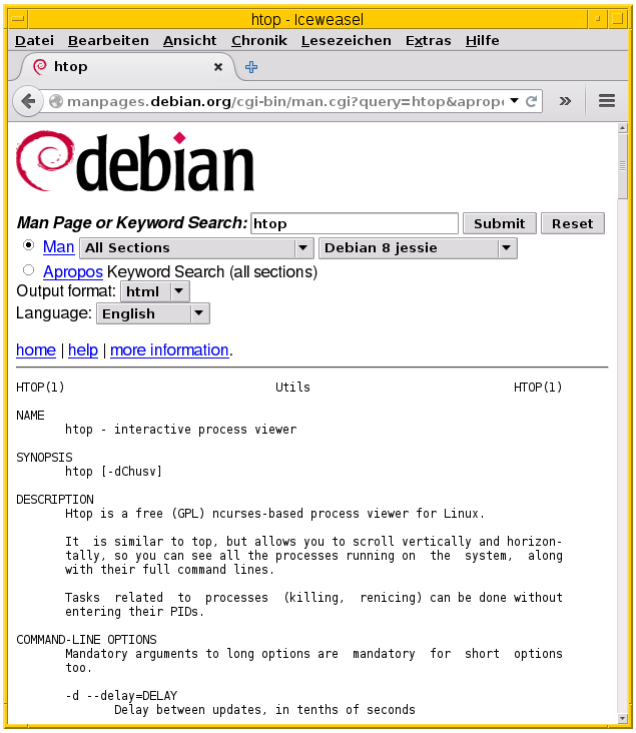

Abbildung 8.16: Suche in der Manpages-Sammlung nach `htop`

Über diesen Service recherchieren Sie in allen Veröffentlichungen von Debian sowie auch *testing*, *unstable* und *experimental*. Über die Auswahllisten legen Sie neben dem zu durchsuchenden Bereich das Ausgabeformat fest — hier HTML, PostScript, PDF oder Plaintext. Der Service ist noch nicht ganz vollständig, so dass derzeit noch nicht alle Manpages für die über das Menü offerierten Sprachen hinterlegt sind.

8.27 Konfigurationsdateien eines Pakets anzeigen

Die meisten Programme verfügen über Konfigurationsdateien, mit denen Sie das jeweilige Programm auf ihre Bedürfnisse individuell einstellen können. Dazu hilft es Ihnen, zu wissen, welche das überhaupt sind und an welcher Stelle sich diese Dateien befinden.

Mit den beiden Werkzeugen `apt-file` und `dlocate` bringen Sie Licht ins Dunkel. Dabei werden jedoch nur die Konfigurationsdateien aufgespürt, die im jeweiligen Paket bereits mitgeliefert werden. Zusätzliche Dateien, die Sie selbst angelegt haben oder im laufenden Betrieb entstanden, kann keines der beiden Werkzeuge erahnen.

Als *Variante 1* nutzen Sie den Aufruf `apt-file show Paketname` und lassen sich den Inhalt des Pakets ausgeben. Ausführlich erklären wir Ihnen das Vorgehen unter Abschnitt 8.23.

Für *Variante 2* nutzen Sie das Programm `dlocate` aus dem gleichnamigen Paket [Debian-Paket-dlocate]. `dlocate` kennt dazu den Schalter `-conf` gefolgt vom Paketnamen. Nachfolgend zeigen wir das für den PDF-Betrachter *xpdf*.

Ermittlung der Konfigurationsdateien zum Paket xpdf mittels dlocate

```
$ dlocate -conf xpdf
/etc/xpdf/xpdfrc
/etc/X11/Xresources/xpdf
$
```

Benötigen Sie zusätzlich die Benutzerrechte der Konfigurationsdateien, deren Besitzer und Eigentümer sowie deren Größe und Zugriffsdatum, ist der Schalter `-lsconf` von großem Nutzen. Dieser bewirkt eine Ausgabe wie das Kommando `ls -la`, wie die nachfolgende Ausgabe deutlich macht:

Ermittlung der Konfigurationsdateien zum Paket xpdf mittels dlocate (ausführliche Ansicht)

```
$ dlocate -lsconf xpdf
-rw-r--r-- 1 root root   43 Aug 21  2010 /etc/X11/Xresources/xpdf
-rw-r--r-- 1 root root 2215 Apr 17  2012 /etc/xpdf/xpdfrc
$
```

8.28 Paketänderungen nachlesen

Für jedes Debianpaket existiert im entsprechenden Quellpaket (siehe Abschnitt 2.7.4) eine Datei mit den erfolgten Änderungen im Paket, ein sogenanntes *Changelog*. Daraus ersehen Sie, was sich im Vergleich zur vorherigen Veröffentlichung getan hat und welche Änderungen der Maintainer zum Originalquellcode vorgenommen hat.

Für APT bietet `apt-get` das Unterkommando `changelog`. Es bezieht das Protokoll der Änderungen für das angefragte Paket direkt vom Paketmirror und gibt dieses im Terminal aus. Die nachfolgenden Ausgaben zeigen den Vorgang für `apt-get` und `aptitude` sowie einen Ausschnitt des Changelogs zum Paket *smartpm*.

Beziehen der Changelog-Informationen zum Paket smartpm

```
$ apt-get changelog smartpm
Holen: 1 Änderungsprotokoll (Changelog) für smartpm (http://packages.debian.org/changelogs/ ↩
    pool/main/s/smart/smart_1.4-2/changelog) [8.573 B]
Es wurden 8.573 B in 1 s geholt (4.754 B/s).
$
```

`aptitude` verfügt dazu ebenfalls über den Schalter `changelog`. Als zusätzlichen Parameter erwartet es den Namen eines Pakets, zudem diese Informationen bezogen und ausgegeben werden. Das angefragte Paket muss dazu nicht auf Ihrem Debiansystem installiert sein.

Die Paketänderungen zum Paket smartpm ausgeben

```
$ aptitude changelog smartpm

smart (1.4-2) unstable; urgency=low

  * Switch to dh_python2 (Thanks to Barry Warsaw)

 -- Free Ekanayaka <freee@debian.org>  Fri, 12 Aug 2011 17:27:20 +0100

smart (1.4-1) unstable; urgency=low
```

```
  * New upstream release
  * Drop several patches (02_fix_fetcher_test, 03_setup,
    06_CVE-2009-3560.patch and 06_CVE-2009-3720.patch) as they were
    all merged upstream

 -- Free Ekanayaka <freee@debian.org>  Tue, 31 May 2011 16:04:52 +0200

...

$
```

8.29 Paket auf Veränderungen prüfen

Installieren Sie ein Debianpaket, landen die darin enthaltenen Dateien üblicherweise eins-zu-eins auf dem Speichermedium. Als Administrator gehört zu Ihren Aufgaben, das System und die Dateien auf Integrität zu prüfen. Das umfasst das Nachschauen, ob die lokalen Dateien aus einem installierten Paket verändert wurden, d.h. ob zwischen der Version vom Paketmirror und der lokalen Version Unterschiede bestehen. Falls ja, ist von Ihnen zu klären, welche Dateien verändert wurden. Es gibt mehrere Situationen, in denen das wichtig ist, gewollte oder unerwünschte Änderungen von Daten festzustellen.

- Welche Unterschiede bestehen zwischen der offiziell verfügbaren Version (und dessen Konfiguration) und den lokalen Einstellungen, sprich: welche Änderungen haben Sie vorgenommen und müssen ggf. bei einer Aktualisierung der Pakete oder des Systems berücksichtigt werden?
- Vorher hat ein anderer Administrator den Rechner betreut. Sie möchten wissen, an welchen Dateien Änderungen von demjenigen vorgenommen wurden.
- Nach einer Reparatur des Dateisystems, bei der zu Paketen gehörende Dateien verändert wurden, prüfen Sie nach, ob die Reparatur erfolgreich war, d.h. ob die Dateien nach wie vor den erwarteten Inhalt haben.

Bei der Klärung dieser Fragen helfen Ihnen u.a. die Werkzeuge `debsums` [Debian-Paket-debsums], `dlocate` [Debian-Paket-dlocate] und auch `dpkg` selbst weiter. Letzteres steht Ihnen mit einem passenden Schalter ab der Version 1.17 ab Debian 8 *Jessie* und Ubuntu 14.04 LTS *Trusty Tahr* zur Verfügung.

8.29.1 MD5-Summen zur Erkennung von Änderungen

Während Debian bei der Verifizierung der bezogenen Pakete auch SHA1- und SHA256-Hashsummen zur kryptografischen Absicherung verwendet (siehe dazu Abschnitt 8.35), werden zum Erkennen von Änderungen an installierten Paketdateien nur MD5-Summen verwendet. Diese sind pro Paket in den Dateien `/var/lib/dpkg/info/*.md5sums` gespeichert. Alle o.g. Programme verwenden die Hashsummen aus diesen via `dpkg` bereitgestellten Dateien.

Die ausschließliche Verwendung von MD5-Summen an dieser Stelle bedeutet, dass diese nicht mehr den heutigen Ansprüchen für das Aufdecken von Datei-Ersetzungen entsprechen, auch wenn diese mit hoher krimineller Energie ausgeführt wurden. Sie können jedoch durchaus helfen, von dilettantischen Einbrechern durchgeführte Datei-Ersetzungen zu finden. Bedenken Sie jedoch dabei, dass die Einbrecher genauso gut auch die o.g. Dateien mit den MD5-Summen angepasst haben könnten. Möchten Sie sich jedoch stärker gegen Datei-Ersetzungen oder Änderungen durch professionelle Angreifer schützen, so reichen die hier genannten Techniken nicht aus. Dazu gibt es spezialisierte Pakete wie z. B. *tripwire*, *samhain*, *aide*, *intergrit*, *fcheck*, *stealth* und *tiger*.

8.29.1.1 MD5-Summen von Dateien mit `dlocate` anzeigen

Mit dem Schalter `-md5sum` des Werkzeugs `dlocate` zeigen Sie die MD5-Summen aller Dateien in einem bestimmten Paket an, so wie sie in o.g. Dateien von `dpkg` gespeichert werden. Nachfolgend sehen Sie die Ausgabe zum Paket *htop*, wobei sich in der linken Spalte die MD5-Summe befindet und in der rechten Spalte die dazugehörige Datei mit ihrem vollständigem Pfad. Die Angaben entsprechen dem Inhalt der Datei `/var/lib/dpkg/info/htop.md5sums`.

Darstellung der MD5-Summen für alle Dateien aus dem Paket htop

```
$ dlocate -md5sum htop
292b696a5b879f1068f7c15073c245cd  usr/bin/htop
194b840f96d3e6bbf29229811a6195c2  usr/share/applications/htop.desktop
75557092070931bcb0fb9a6d74575542  usr/share/doc/htop/AUTHORS
0c9303726b090f478b383dd059b3265f  usr/share/doc/htop/README
3adf8fa10448f27bb30385b37eb14231  usr/share/doc/htop/changelog.Debian.gz
84555fa6bc74568aea8de2a18072d5b2  usr/share/doc/htop/changelog.gz
ee7657b42989a83c9b04a179b35e59e1  usr/share/doc/htop/copyright
58a889c99141c2945c1c50bb51d314c6  usr/share/man/man1/htop.1.gz
f059e3f0159a5aeb761d41514a117310  usr/share/menu/htop
5bbd19dc6cccaf0a74866a92f5cca75c  usr/share/pixmaps/htop.png
$
```

8.29.2 Dateien paketbezogen mit `dlocate` überprüfen

`dlocate` kann nicht nur die MD5-Summe für eine Datei ausgeben, sondern diese auch überprüfen. Dazu benutzen Sie den Schalter `-md5check`. Falls die ermittelte MD5-Summe mit dem Original aus dem Paket übereinstimmt, ergänzt `dlocate` hinter dem Dateinamen ein `OK`, andernfalls ein `FAILED`.

Bitte beachten Sie dabei, dass `dlocate -md5check` keine Konfigurationsdateien überprüft und auch nur die Dateien von explizit angegebenen Paketen überprüfen kann.

Überprüfung der MD5-Summen für jede einzelne Datei aus dem Paket htop

```
$ dlocate -md5check htop
usr/bin/htop: OK
usr/share/applications/htop.desktop: OK
usr/share/doc/htop/AUTHORS: OK
usr/share/doc/htop/README: OK
usr/share/doc/htop/changelog.Debian.gz: OK
usr/share/doc/htop/changelog.gz: OK
usr/share/doc/htop/copyright: OK
usr/share/man/man1/htop.1.gz: OK
usr/share/menu/htop: OK
usr/share/pixmaps/htop.png: OK
$
```

8.29.3 Dateien überprüfen mit `debsums`

Genauso wie `dlocate` kann auch `debsums` die Dateien eines Pakets auf Integrität überprüfen. Dazu braucht es jedoch keine weitere Option, da das Überprüfen von Dateien die einzige Aufgabe von `debsums` ist:

debsums beim Prüfen des Pakets htop

```
$ debsums htop
/usr/bin/htop                                   OK
/usr/share/applications/htop.desktop            OK
/usr/share/doc/htop/AUTHORS                     OK
/usr/share/doc/htop/README                      OK
/usr/share/doc/htop/changelog.Debian.gz         OK
/usr/share/doc/htop/changelog.gz                OK
/usr/share/doc/htop/copyright                   OK
/usr/share/man/man1/htop.1.gz                   OK
/usr/share/menu/htop                            OK
/usr/share/pixmaps/htop.png                     OK
$
```

Im Gegensatz zu `dlocate` braucht `debsums` jedoch nicht notwendigerweise einen Paketnamen als Parameter. Rufen Sie das Werkzeug `debsums` ohne weitere Parameter auf, so prüft es alle Dateien (außer Konfigurationsdateien in `/etc/`) sämtlicher installierten Pakete auf Veränderungen zum Original und gibt hinter dem Dateinamen den Wert `OK` für unverändert und `FAILED` für modifizierte Daten aus. Dieser Schritt eignet sich gut, um ihr gesamtes System einer Integritätsprüfung zu unterziehen.

debsums bei der Arbeit

```
# debsums
/usr/bin/a2ps                          OK
/usr/bin/a2ps-lpr-wrapper              OK
/usr/bin/card                          OK
/usr/bin/pdiff                         OK
/usr/bin/psmandup                      OK
/usr/bin/psset                         OK
/usr/bin/texi2dvi4a2ps                 OK
/usr/share/a2ps/README                 OK
/usr/share/a2ps/afm/fonts.map          OK
...
#
```

Desweiteren hat `debsums` noch ein paar nützliche Schalter:

`-a` (Langform `--all`)
: Überprüfung aller Dateien.

`-c` (Langform `--changed`)
: Nur die Dateien anzeigen, die sich geändert haben.

Auflistung der Dateien, die sich geändert haben

```
# debsums --changed
/usr/local/Brother/Printer/HL2250DN/inf/brHL2250DNfunc
/usr/local/Brother/Printer/HL2250DN/inf/brHL2250DNrc
debsums: missing file /usr/share/doc/hl2250dnlpr/copyright (from hl2250dnlpr package)
debsums: missing file /usr/share/doc/hl2250dnlpr/changelog.Debian.gz (from hl2250dnlpr ↩
    package)
debsums: missing file //opt/PDFStudio/jre/lib/charsets.jar.pack (from pdfstudio package)
#
```

`-e` (Langform `--config`)
: Überprüfung der *Conffiles*. *Conffiles* sind Konfigurationsdateien, die vom Paket ausgeliefert werden und somit vorab deklariert wurden. Diese befinden sich fast immer unterhalb des Verzeichnisses `/etc/`.

Auflistung aller Conffiles des Pakets unburden-home-dir mit Zustand:

```
$ debsums -e unburden-home-dir
/etc/unburden-home-dir.list                FAILED
/etc/unburden-home-dir                         OK
/etc/default/unburden-home-dir             FAILED
/etc/X11/Xsession.d/95unburden-home-dir        OK
$
```

Möchten Sie nur die Konfigurationsdateien (genauer *Conffiles*) eines Pakets auflisten, die lokal geändert wurden, so kombinieren Sie die beiden Schalter `-c` und `-e` miteinander:

Auflistung geänderter Conffiles des Pakets unburden-home-dir

```
$ debsums -ce unburden-home-dir
/etc/default/unburden-home-dir
/etc/unburden-home-dir.list
$
```

Möchten Sie die Originaldatei wiedereinspielen (und damit die Änderungen rückgängig machen), ermitteln Sie zuerst das Paket, in dem besagte Datei enthalten ist (siehe Abschnitt 8.22) und installieren dieses dann erneut (siehe Abschnitt 8.37).

Bitte beachten Sie, dass das bei *Conffiles* nicht funktioniert, da `dpkg` nur dann wegen geänderter (oder gelöschter) Konfigurationsdateien fragt, wenn sich die Konfigurationsdatei auch im Paket geändert hat. Dies ist bei einer Reinstallation nie der Fall. Hier hilft entweder, die Datei aus dem heruntergeladenen Paket manuell zu extrahieren oder zunächst das Paket mit *dpkg --purge* vollständig zu entfernen und danach wieder zu installieren.

Bei der Benutzung von `debsums` spielen die Berechtigungen des Benutzers eine Rolle. Die Integrität von Dateien, die für normale Benutzer nicht lesbar sind, können nur vom Benutzer *root* geprüft werden.

Auflistung geänderter Conffiles des Pakets sudo geht nur root-Rechten:

```
$ debsums -e sudo
/etc/pam.d/sudo                                     OK
/etc/init.d/sudo                                    OK
debsums: can't open sudo file /etc/sudoers (Permission denied)
debsums: can't open sudo file /etc/sudoers.d/README (Permission denied)
$ sudo debsums -e sudo
/etc/pam.d/sudo                                     OK
/etc/sudoers                                        OK
/etc/init.d/sudo                                    OK
/etc/sudoers.d/README                               OK
$
```

8.29.4 Dateien mit `dpkg -V` überprüfen

Ab *dpkg* Version 1.17 kann auch `dpkg` selbst Dateien anhand der gespeicherten MD5-Summen auf Unversehrtheit überprüfen. Im Gegensatz zu `debsums` und `dlocate -md5check` überprüft es *Conffiles* stets mit und zeigt auch immer nur Dateien an, die sich nicht mehr im Originalzustand befinden.

Die passende Option dazu ist `-V` bzw. in der Langform `--verify`. Geben Sie zum Aufruf einen oder mehrere Paketnamen als Parameter mit, so werden nur die Dateien dieser Pakete überprüft:

Dateien der Pakete unburden-home-dir und ack-grep mit dpkg -V überprüfen

```
$ dpkg -V unburden-home-dir ack-grep
??5?????? c /etc/unburden-home-dir.list
??5?????? c /etc/default/unburden-home-dir
??5??????   /usr/bin/ack
$
```

Das Ausgabeformat stellen Sie über die Option `--verify-format` ein. Das Standardformat ist von *RPM* übernommen [Bailey-Maximum-RPM-verify]. Da `dpkg` bisher nur die MD5-Summe überprüft, werden alle anderen Spalten nur als Fragezeichen ausgegeben. Erscheint ein einzelnes *c* in der Ausgabe, handelt es sich hierbei um *Conffiles*.

8.30 Liste der zuletzt geänderten Abhängigkeiten

Um herauszufinden, welche Abhängigkeiten eines Pakets sich zuletzt geändert haben und ob vielleicht dabei ein Bug zum Vorschein kam, sparen Sie sich mit dem Programm `which-pkg-broke` aus dem Paket *debian-goodies* [Debian-Paket-debian-goodies] einiges an Arbeit. Übersetzt heißt der Programmname sinngemäß „welches Paket machte [es] kaputt“. Nachfolgende Übersicht zeigt Ihnen die zuletzt geänderten Abhängigkeiten von `apt`.

Anzeige der zuletzt geänderten Abhängigkeiten von apt

```
$ which-pkg-broke apt
libacl1:amd64                              Tue Apr  8 18:57:57 2014
libattr1:amd64                             Tue Apr  8 18:57:58 2014
liblzma5:amd64                             Tue Apr  8 18:58:11 2014
tar                                        Tue Apr  8 18:58:20 2014
```

```
zlib1g:amd64                            Tue Apr  8 18:58:23 2014
debian-archive-keyring                  Tue Apr  8 18:58:41 2014
readline-common                         Tue Apr  8 18:58:59 2014
libreadline6:amd64                      Tue Apr  8 18:58:59 2014
libselinux1:amd64                       Fri May 16 19:31:14 2014
install-info                            Tue Jun  3 14:02:14 2014
dpkg                                    Thu Jun  5 23:50:19 2014
libusb-0.1-4:amd64                      Fri Jul  4 02:00:58 2014
gpgv                                    Tue Jul  8 00:19:12 2014
gnupg                                   Tue Jul  8 00:19:15 2014
libapt-pkg4.13:amd64                    Sat Jul 12 02:37:23 2014
apt                                     Sat Jul 12 02:37:26 2014
libc6:amd64                             Sun Jul 13 13:09:04 2014
multiarch-support                       Sun Jul 13 13:09:43 2014
libtinfo5:amd64                         Sun Jul 20 13:39:10 2014
libpcre3:amd64                          Thu Jul 24 09:45:03 2014
gcc-4.9-base:amd64                      Thu Jul 31 18:11:34 2014
libgcc1:amd64                           Thu Jul 31 18:11:36 2014
libstdc++6:amd64                        Thu Jul 31 18:11:36 2014
libbz2-1.0:amd64                        Fri Aug  1 14:45:59 2014
$
```

Die Ausgabe umfasst in der linken Spalte den Paketnamen (siehe Abschnitt 2.11) und ggf. die Architektur (siehe Abschnitt 1.2) sowie in der rechten Spalte den Zeitpunkt der erfolgten Änderung. Sie ersehen daraus, welche der Abhängigkeiten (hier am Beispiel von `apt`) zu welchem Zeitpunkt zuletzt auf diesem System aktualisiert wurden. Wenn Sie jetzt noch wissen, wann der zu lokalisierende Fehler zuerst bemerkbar wurde, schränken Sie über die Datumsangaben recht schnell ein, welches Paket den Fehler verursacht hat.

8.31 Paketdatei nur herunterladen

APT und `aptitude` sind dafür gedacht, Softwarepakete vom Paketmirror zu beziehen und diese danach sofort auf ihrem System einzuspielen. Es besteht natürlich auch die Möglichkeit, diesen Vorgang in die beiden Einzelschritte zu zerlegen, d.h. Herunterladen des Pakets und die Installation aus dem Paketcache (siehe Abschnitt 8.32). Mehrere Programme mit unterschiedlichen Schaltern bieten sich dafür an.

Für das herunterladen und speichern bestehen zwei Modi – das Speichern im aktuellen Verzeichnis (Modus 1) und im Paketcache (Modus 2). Für beides existieren verschiedene Unterkommandos und Aufrufmöglichkeiten.

Für den Modus 1 akzeptieren `apt-get` und `aptitude` das Unterkommando `download`. Das Paket *bash-doc* beinhaltet die Dokumentation zur Bash – ein Bezug des Pakets via `apt-get` liefert Ihnen das folgende Ergebnis:

Bezug des Pakets bash-doc via apt-get und Speicherung im lokalen Verzeichnis

```
# apt-get download bash-doc
Holen: 1 Herunterladen von bash-doc 4.2+dfsg-0.1 [696 kB]
Es wurden 696 kB in 0 s geholt (1.549 kB/s).
# ls bash-doc_4.2+dfsg-0.1_all.deb -la
-rw-r--r-- 1 root root 696268 Dez 30  2012 bash-doc_4.2+dfsg-0.1_all.deb
#
```

Dieser Aufruf ist identisch mit der Benutzung eines Webbrowsers. Dazu wählen Sie beispielsweise im Debian-Paketarchiv das gewünschte Paket aus und legen es über Datei speichern unter in einem lokalen Verzeichnis ab (siehe Abbildung 8.17).

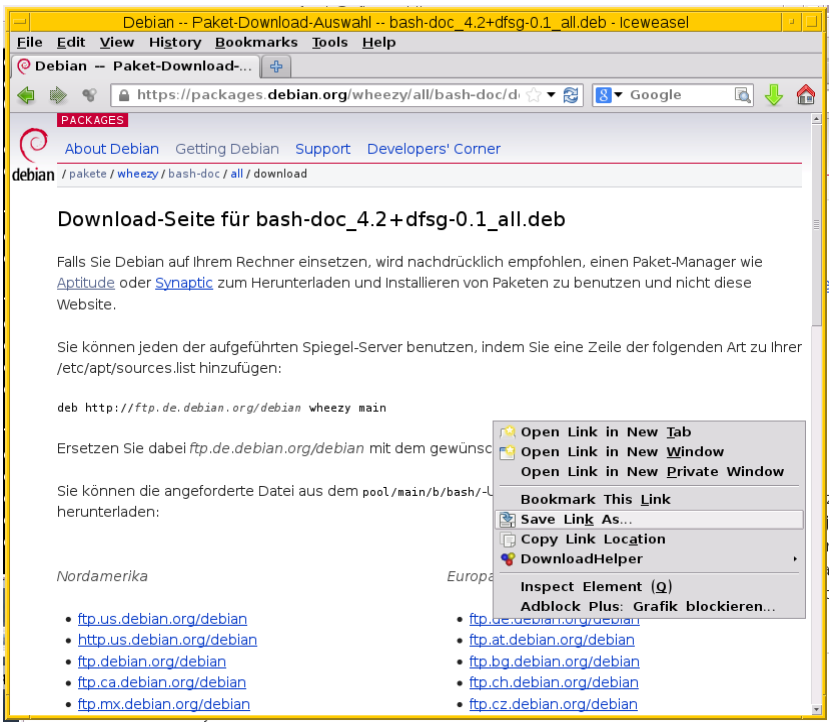

Abbildung 8.17: Bezug des Pakets *bash-doc* über den Webbrowser

Der Modus 2 kommt zum Zug, wenn Sie das Paket hingegen im lokalen Paketcache (siehe Kapitel 7) abspeichern möchten. Dazu verstehen `apt-get` und `aptitude` zum Unterkommando `install` die Option `-d` (Langform `--download-only`). Nachfolgende Ausgabe zeigt, wie sich `aptitude` dabei verhält. Das Paket wird nicht installiert, sondern im Paketcache unter `/var/cache/apt/archives/` abgespeichert, sofern es vollständig bezogen wurde. Nur teilweise heruntergeladene Pakete liegen hingegen unter `/var/cache/apt/archives/partial/`.

Bezug des Pakets bash-doc via aptitude und Speicherung im Paketcache

```
# aptitude --download-only install bash-doc
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  bash-doc
0 Pakete aktualisiert, 1 zusätzlich installiert, 0 werden entfernt und 16 nicht  ↩
    aktualisiert.
696 kB an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 1.430 kB  ↩
    zusätzlich belegt sein.
Holen: 1 http://ftp.de.debian.org/debian/ wheezy/main bash-doc all 4.2+dfsg-0.1 [696 kB]
696 kB wurden in 0 s heruntergeladen (1.761 kB/s)
#
```

Die Verwendung beider Modi ist sinnvoll, wenn Sie beispielsweise eine Installation vorbereiten und dabei im Vorfeld überprüfen möchten, ob alles reibungslos funktioniert. Ebenso zählt das Ausprobieren dazu – das Schauen, was passiert, ohne eine tatsächliche Veränderung des Paketbestands auf dem System vorzunehmen.

Ein weiterer Fall ist die Aktualisierung von Paketen ohne Internetzugang (siehe auch Kapitel 37). Damit können Sie vorab bereits alle Pakete zusammenstellen, die Sie im Bedarfsfall benötigen und diese dann aus dem Paketcache oder aus einem lokalen Verzeichnis installieren, ohne auf eine bestehende Netzverbindung angewiesen zu sein.

8.32 Installation zwischengespeicherter Pakete aus dem Paketcache

Liegt das Paket bereits oder noch im Paketcache, kann APT dieses von dort entnehmen und sofort installieren. Ein Bezug vom Paketmirror ist in diesem Fall nicht mehr erforderlich und spart sowohl Zeit, als auch Bandbreite. Dazu benötigt `apt-get`

die Option `--no-download` zum Unterkommando `install`. Nachfolgende Ausgabe zeigt das anhand des Pakets *bash-doc*, welches bereits im Paketcache liegt.

Installation des Pakets bash-doc via apt-get aus dem Paketcache

```
# apt-get --no-download install bash-doc
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden NEUEN Pakete werden installiert:
  bash-doc
0 aktualisiert, 1 neu installiert, 0 zu entfernen und 16 nicht aktualisiert.
Es müssen noch 0 B von 696 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 1.430 kB Plattenplatz zusätzlich benutzt.
Laden der Fehlerberichte ... Erledigt
»Found/Fixed«-Informationen werden ausgewertet ... Erledigt
Vormals nicht ausgewähltes Paket bash-doc wird gewählt.
(Lese Datenbank ... 299513 Dateien und Verzeichnisse sind derzeit installiert.)
Entpacken von bash-doc (aus .../bash-doc_4.2+dfsg-0.1_all.deb) ...
bash-doc (4.2+dfsg-0.1) wird eingerichtet ...
#
```

Paket vom Paketmirror beziehen und automatisch (sofort) installieren
Um eines oder mehrere Pakete sofort zu installieren, kennen APT und `aptitude` das Unterkommando `install`. Alle vollständig bezogenen Pakete landen danach automatisch im Paketcache unter `/var/cache/apt/archives`. Ausführlich gehen wir darauf unter Pakete installieren in Abschnitt 8.36 und Paketcache wieder aufräumen in Abschnitt 7.3 ein.

8.33 Sourcepakete beziehen

Die Möglichkeit, auch die Quellpakete (siehe Abschnitt 2.7.4) zu den verwendeten Programmen zu erhalten, zählt zu den zentralen Säulen Freier Software. Neben dem Lerneffekt steht die Befriedigung der Neugierde, zu sehen, woraus überhaupt ein Debian-Binärpaket (siehe Abschnitt 2.7.1) entsteht und aus welchen Komponenten sich dieses zusammensetzt.

Damit erhalten Sie einen Blick hinter die Kulissen und können anhand des Quellcodes ersehen, wie die Software programmiert wurde. Nur über diesen Schritt können Sie ganz konkret nachvollziehen, wie diese funktioniert. Das hilft Ihnen insbesondere auch dabei, die Ursache zu lokalisieren, wenn ein Programm sich entgegen ihrer Erwartungen verhält.

Viele Entwickler weisen der Dokumentation ihrer Software häufig einen niedrigen Stellenwert zu. Es kommt daher vor, dass die Dokumentation unvollständig, fehlerhaft bzw. veraltet ist oder in einer Sprache vorliegt, die sie nicht beherrschen. Schwachpunkte sind zudem die Verfahren, welche implementiert wurden, aber auch die Parameter, Schalter und Konfigurationsdateien, mit der Sie das Verhalten der Software steuern und beeinflussen können.

Das Programm `apt-get` bringt hier den Schalter `source` mit und erwartet danach die Angabe eines oder mehrerer Paketnamen. Damit `apt-get` nach dem Aufruf die Quellpakete auch beziehen kann, benötigt es einen entsprechenden Eintrag in der Liste der Paketquellen (siehe Abschnitt 3.3). Für die Veröffentlichung Debian 8 Jessie sieht der Eintrag wie folgt aus:

```
deb-src http://ftp.de.debian.org/debian/ jessie main contrib non-free
```

APT wertet die Paketbeschreibung aus, bezieht danach alle Quellpakete von dem angegebenen Paketmirror – den Debian Source Code (`dsc`) plus Paketierung (siehe Abschnitt 4.2.2) – , aus denen das Binärpaket zusammengebaut wurde und überprüft diese Komponenten (siehe Abschnitt 8.35) anhand deren öffentlichem Schlüssel. Am Schluss werden die drei Archive `dsc`, `tar` und `diff` im aktuellen Verzeichnis entpackt.

Gibt es zusätzliche Änderungen am Quellcode in Form von Patches, werden diese ebenfalls bezogen und nacheinander auf den entpackten Quellcode angewendet. Nachfolgendes Beispiel zeigt diesen Vorgang anhand des Pakets *libapache2-mod-authn-yubikey* für den Webserver Apache:

Bezug des Sourcepakets libapache2-mod-authn-yubikey mit APT

```
$ apt-get source libapache2-mod-authn-yubikey
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Es müssen 22,5 kB an Quellarchiven heruntergeladen werden.
Holen: 1 http://ftp.de.debian.org/debian/ wheezy/main libapache2-mod-authn-yubikey 1.0-1 ( ↩
    dsc) [1.891 B]
Holen: 2 http://ftp.de.debian.org/debian/ wheezy/main libapache2-mod-authn-yubikey 1.0-1 ( ↩
    tar) [16,5 kB]
Holen: 3 http://ftp.de.debian.org/debian/ wheezy/main libapache2-mod-authn-yubikey 1.0-1 ( ↩
    diff) [4.115 B]
Es wurden 22,5 kB in 5 s geholt (4.095 B/s).
gpgv: Schlüsselblockhilfsmittel‘/home/frank/.gnupg/trustedkeys.gpg’: Fehler beim Öffnen der ↩
     Datei
gpgv: Unterschrift vom Do 17 Feb 2011 16:22:26 CET mittels RSA-Schlüssel ID 8649AA06
gpgv: Unterschrift kann nicht geprüft werden: Öffentlicher Schlüssel nicht gefunden
dpkg-source: Warnung: Fehler beim Überprüfen der Signatur von ./libapache2-mod-authn- ↩
    yubikey_1.0-1.dsc
dpkg-source: Information: libapache2-mod-authn-yubikey wird nach libapache2-mod-authn- ↩
    yubikey-1.0 extrahiert
dpkg-source: Information: libapache2-mod-authn-yubikey_1.0.orig.tar.bz2 wird entpackt
dpkg-source: Information: libapache2-mod-authn-yubikey_1.0-1.debian.tar.gz wird entpackt
$
```

Desweiteren existiert auch eine Alternative namens `dget` aus dem Paket *devscripts* [Debian-Paket-devscripts]. Darüber kombinieren Sie den Bezug der Paketinhalte. `dget` akzeptiert als Parameter eine Liste von Paketnamen, die es nacheinander vom Paketmirror bezieht und im lokalen Verzeichnis speichert. Liegt das betreffende Paket bereits im Paketcache, entnimmt `dget` dieses von dort [Kemp-dget].

Bezug der beiden Pakete bash-doc und libbash-doc via dget

```
# dget bash-doc
dget: using /var/cache/apt/archives/bash-doc_4.2+dfsg-0.1_all.deb (copy)
#
# dget libbash-doc
dget: retrieving http://ftp.de.debian.org/debian/pool/main/libb/libbash/libbash-doc_0 ↩
    .9.11-1_all.deb
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 20468  100 20468    0     0   166k      0 --:--:-- --:--:-- --:--:--  243k
#
```

8.34 Sourcepakete anzeigen

Zur Philosophie der Freien Software gehört der vollständige Zugang zu den Quelldaten der Binärpakete. Das Werkzeug `apt-cache` ermöglicht es Ihnen, über das Unterkommando `showsrc` alle Informationen zu einem bestimmten Quellpaket anzuzeigen, welches in Debian verfügbar ist.

Die nachfolgende Ausgabe zeigt das Ergebnis zum Paket *htop* an. Neben dem Paketnamen für das Binär- und Quellpaket (`Binary` und `Package`) sehen Sie die Version (`Version`) und den Maintainer (`Maintainer`) sowie das Paketformat (`Format`) und die Architektur (`Architecture`), für welche das vorliegende Paket übersetzt werden kann. Neben dem Schlüsselwort `Build-Depends` sind alle Pakete samt deren Version aufgeführt, die zum Übersetzen des Programmcodes erforderlich sind. Unter `Files` sind zudem noch alle Dateien samt deren Hashwert (`Checksums-Sha1` und `Checksums-Sha256`) benannt. Den Abschluß der Beschreibung bilden die Projektwebseite (`Homepage`), die Paketliste (`Package-List`), das Verzeichnis auf dem Paketmirror (`Directory`), die Paketpriorität (`Priority`) und die Einsortierung in die Paketkategorie (`Section`).

Ausgabe der Informationen zum Sourcepaket zu htop

```
$ apt-cache showsrc htop
Package: htop
Binary: htop
Version: 1.0.1-1
Maintainer: Eugene V. Lyubimkin <jackyf@debian.org>
Build-Depends: debhelper (>= 7), libncurses5-dev, libncursesw5-dev, autotools-dev, quilt  ↩
    (>= 0.40), python-minimal, libhwloc-dev [!linux-any]
Architecture: any
Standards-Version: 3.9.2
Format: 1.0
Files:
 fbaa099edb84fd7ea95fa41d4bf43852 1112 htop_1.0.1-1.dsc
 d3b80d905a6bff03f13896870787f901 384683 htop_1.0.1.orig.tar.gz
 5952c54e78d6147adbdd541764491796 9113 htop_1.0.1-1.diff.gz
Checksums-Sha1:
 3c3eb973c4399fd24c578643790de158b39fe87e 1112 htop_1.0.1-1.dsc
 bad226ec887a2b7ea5042879ed18e067812d030e 384683 htop_1.0.1.orig.tar.gz
 63306ced4fa534698fc8e111035fc5cbdfe35ab2 9113 htop_1.0.1-1.diff.gz
Checksums-Sha256:
 2b80e492eac78607fd6962c88823e1be537e800f293189d02ede5ef5ad8994e4 1112 htop_1.0.1-1.dsc
 07db2cbe02835f9e186b9610ecc3beca330a5c9beadb3b6069dd0a10561506f2 384683 htop_1.0.1.orig. ↩
    tar.gz
 d3b0b9edd356cd3078ac582ebeda20bd5972bc2ee903e766c4adf4ab5c61d249 9113 htop_1.0.1-1.diff.gz
Homepage: http://htop.sourceforge.net
Package-List:
 htop deb utils optional
Directory: pool/main/h/htop
Priority: source
Section: utils
$
```

8.35 Bezogenes Paket verifizieren (GPG-Key)

Installieren Sie ein Paket von einer Paketquelle, greifen eine ganze Reihe von automatischen Überprüfungen, die sicherstellen, dass die von Ihnen bezogenen Daten vertrauenswürdig sind. Mit den nachfolgend beschriebenen Vorgehensweisen können Sie die entsprechenden Einzelschritte selbst nachvollziehen und durchführen.

Diese Überprüfungen stellen sicher, dass ein von Ihnen bezogenenes Debianpaket dem Paketmirror so entnommen wurde, wie es von der Distribution in der Veröffentlichung zur Verfügung gestellt wurde. Sie schließen damit aus, dass zwischenzeitlich Veränderungen von einer dritten Partei oder auf dem Paketmirror stattgefunden haben. Weiterhin wissen Sie danach ebenfalls, dass das Paket vollständig zu Ihnen übertragen wurde und dass das Paket auf dem Weg vom Paketmirror zu Ihnen nicht verändert wurde. U.a. bilden diese Schritte die Vertrauensbasis für die von Ihnen bezogene Software.

Aus diesem Grund sind sowohl die offiziellen Paketquellen von Debian, als auch die darüber referenzierten Quellpakete signiert (siehe dazu Abschnitt 3.12). Die Signaturen werden mithilfe der Public-Key-Kryptographie erststellt. Mit dem entsprechenden öffentlichen Schlüssel (engl. *Public Key*) können Sie (und jeder andere Benutzer ebenso) überprüfen, ob das Paket vertrauenswürdig ist.

8.35.1 Basis

Die Grundlage dazu bildet die Vertrauenskette bei Debian, die sich vom Entwickler zum Build-Daemon (kurz „buildd“) bis hin zum FTP-Master-Server, den Paketlisten, dem Debian-Archive-Keyring und dem Debian-Keyring erstreckt. Genutzt wird dabei eine Kombination aus kryptographischen Hashsummen und einer digitalen Signatur.

Auf den Debian-Spiegelservern befindet sich pro Veröffentlichung eine digital signierte Datei namens `Release`. Sie beinhaltet die Namen der Paketlisten (heutzutage meist `Packages`, `Packages.gz` und `Packages.xz`, früher oft auch noch `Packa`

ges.bz2) sowie deren Hashsummen als MD5-, SHA1- und SHA256-Variante. Mit der digitalen Signatur der Release-Datei und den darin enthalten Hashsummen wird sichergestellt, dass diese Dateien nicht verändert wurden.

Die Datei Packages (wie auch deren komprimierten Varianten) beinhaltet wiederum eine Liste von Paketen bzw. deren Dateien, die für diese Veröffentlichung zur Verfügung stehen – und deren Hash-Summen. Dies stellt wiederum sicher, dass die Paketdateien aus der Liste nicht verändert wurden.

Durch die gesamte Kette aus Paket-Hashsummen in den Paketlisten und Paketliste-Hashsummen in der Release-Datei garantiert die einzelne digitale Signatur auf der Release-Datei die Integrität sämtlicher Pakete einer Veröffentlichung.

Vertrauenswürdige Schlüssel verwalten Sie mit dem Programm apt-key aus dem Paket *apt*. Dazu gehört ein Schlüsselring von öffentlichen GnuPG-Schlüsseln, sog. „Public Keys“, mit denen die Signaturen in der Datei Release.gpg auf den Debian-Spiegelservern überprüft werden können. Dieser Schlüsselring ist im Paket *debian-archive-keyring* enthalten und lokal in Dateien im Verzeichnis /etc/apt/trusted.gpg.d gespeichert.

Dazu kommt noch die Datei /etc/apt/trusted.gpg, welche heutzutage nur noch manuell per apt-key add hinzugefügte Schlüssel enthält. Bei früheren Veröffentlichungen war diese Datei die alleinige Quelle zur Überprüfung von Veröffentlichungssignaturen.

Die Signaturen der von Ihnen verwendeten Paketquellen zeigen Sie mit dem Aufruf apt-key finger'{empty}footnote:[Da die Datei `/etc/apt/trusted.gpg teilweise für normale User nicht lesbar ist, kann es sein, dass Sie dieses Kommando mit Root-Rechten ausführen müssen.] an. Nachfolgend sehen Sie beispielhaft die Signaturen zum Opera Software Archive, dem Mendeley Desktop Team und dem Debian Archive für die beiden Veröffentlichungen *Wheezy* und *Jessie*.

Liste der Signaturen (Ausschnitt)

```
# apt-key finger
/etc/apt/trusted.gpg
- --------------------
pub   1024D/30C18A2B 2012-10-29 [verfallen: 2014-10-29]
  Schl.-Fingerabdruck = ABCD 165A F57C AC92 18D2  872B E585 066A 30C1 8A2B
uid                  Opera Software Archive Automatic Signing Key 2013 <packager@opera.com>

pub   2048R/6F036044 2011-02-21
  Schl.-Fingerabdruck = 26BB 0219 1EF4 588D 3A7B  C30F D800 C7D6 6F03 6044
uid                  Mendeley Desktop Team <desktop@mendeley.com>
sub   2048R/F9CE0BFD 2011-02-21

/etc/apt/trusted.gpg.d/debian-archive-jessie-stable.gpg
- ------------------------------------------------------
pub   4096R/518E17E1 2013-08-17 [verfällt: 2021-08-15]
  Schl.-Fingerabdruck = 75DD C3C4 A499 F1A1 8CB5  F3C8 CBF8 D6FD 518E 17E1
uid                  Jessie Stable Release Key <debian-release@lists.debian.org>

/etc/apt/trusted.gpg.d/debian-archive-wheezy-automatic.gpg
- ---------------------------------------------------------
pub   4096R/46925553 2012-04-27 [verfällt: 2020-04-25]
  Schl.-Fingerabdruck = A1BD 8E9D 78F7 FE5C 3E65  D8AF 8B48 AD62 4692 5553
uid                  Debian Archive Automatic Signing Key (7.0/wheezy) <ftpmaster@debian. ↩
    org>

/etc/apt/trusted.gpg.d/debian-archive-wheezy-stable.gpg
- ------------------------------------------------------
pub   4096R/65FFB764 2012-05-08 [verfällt: 2019-05-07]
  Schl.-Fingerabdruck = ED6D 6527 1AAC F0FF 15D1  2303 6FB2 A1C2 65FF B764
uid                  Wheezy Stable Release Key <debian-release@lists.debian.org>

#
```

TODO: gui-apt-key/curses-apt-key doppelt gemoppelt: Auch schon in Abschnitt 3.12 aufgeführt.

Für eine graphische Darstellung nutzen Sie stattdessen das Werkzeug gui-apt-key aus den gleichnamigen Paket [Debian-Paket-gui-apt-key]. Es existiert auch eine TUI-Variante namens curses-apt-key [curses-apt-key]. Diese ist aber zur Zeit

noch nicht offizieller Bestandteil von Debian, da dazu zuerst das Paket `gui-apt-key` in Bibliothek und Frontend aufgespalten werden müsste. Details dazu finden Sie im Debian-Bugtracker [curses-apt-key-braucht-gui-apt-key-aufsplittung]. In Abbildung 8.18 sehen Sie `gui-apt-key` mit der Überprüfung des (inzwischen abgelaufenen) Schlüssels für das Opera Software Archive.

Abbildung 8.18: Informationen zu einem ausgewählten Schlüssel in `gui-apt-key` anzeigen

8.35.2 Nur ein Einzelpaket prüfen

- `apt-key` (siehe [Debian-Wiki-SecureApt])
- empfohlene Vorgehensweise zur Validierung
  - ToDo
- was passiert, wenn das Paket keinen gültigen GPG-Key hat
  - wie erkenne ich das bzw. bekomme das heraus
  - Ausgabe einer Warnung (W) bei `apt-get` und `aptitude` (siehe Abbildung 8.19)

Aktualisierung der Paketlisten mit erkanntem GPG-Fehler

```
# apt-get update
...
Hole:10 http://deb.opera.com squeeze/non-free i386 Packages [774 B]
Es wurden 1.250 kB in 3 s geholt (329 kB/s)
Paketlisten werden gelesen... Fertig
W: GPG-Fehler: http://deb.opera.com squeeze Release: Die folgenden Signaturen konnten
nicht überprüft werden, weil ihr öffentlicher Schlüssel nicht verfügbar ist:
NO_PUBKEY E585066A30C18A2B
#
```

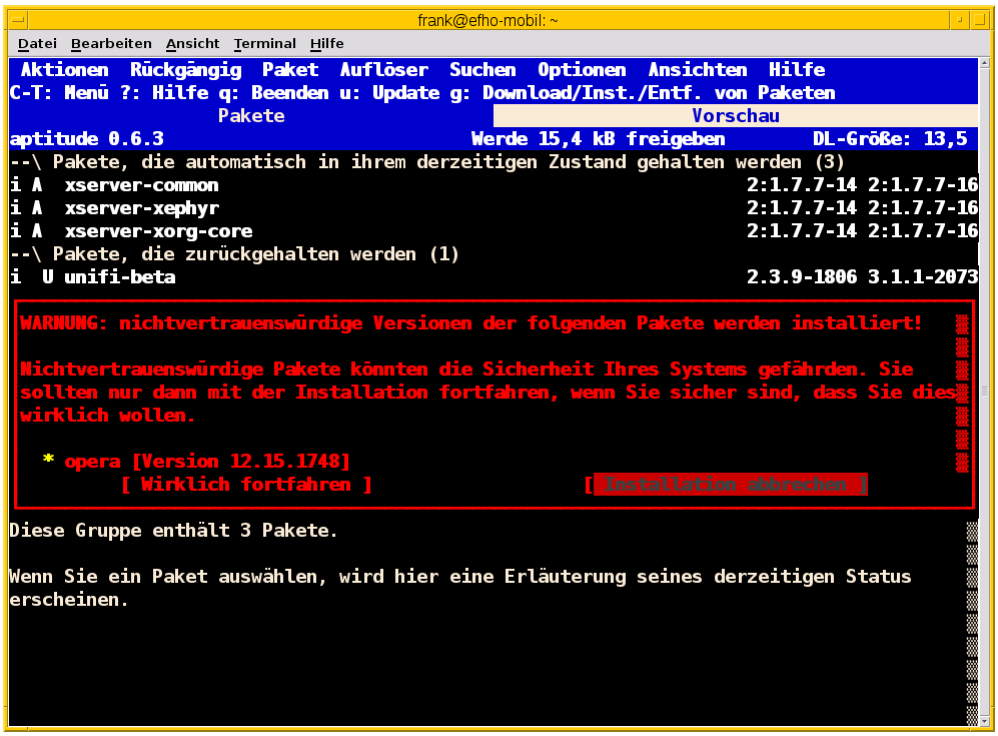

Abbildung 8.19: Ausgabe einer *deutlichen* Warnung bei `aptitude`

8.35.3 Alle bereits installierten Pakete und Dateien prüfen

Um eine ganze Installation auf Korrektheit und bzgl. möglicher Veränderungen zu überprüfen, müssen Sie nicht jedes Paket einzeln anschauen. Wie Sie in dieser Situation vorgehen, lesen Sie unter Abschnitt 8.29 nach.

8.36 Pakete installieren

Die Installation von Paketen und die dazugehörigen Aufrufe gehören aus unserer Sicht zu den Aktionen bei der Paketverwaltung, welche am häufigsten genutzt werden. Nachfolgend beschreiben wir, mit welchen Aufrufen Sie Pakete vom Paketmirror beziehen und danach sofort auf Ihrem System installieren. Wie Sie die Paketdateien nur herunterladen, ohne diese zu installieren, lesen Sie in Abschnitt 8.31.

Für APT und `aptitude` lässt sich der Vorgang mit „Aufruf von `dpkg -i Paketname` in der richtigen Reihenfolge“ umschreiben. Dabei erfolgt zudem die Beachtung der jeweiligen Paketabhängigkeiten – noch fehlende und zusätzlich benötigte Pakete werden erkannt und vom Paketmirror mitbezogen. Die Voraussetzung, dass das angegebene Paket bereits im (lokalen) Verzeichnis liegt, muss nicht erfüllt sein.

Wir empfehlen Ihnen, `dpkg` nur im Ausnahmefall zu benutzen. Der Umgang mit `dpkg` bzw. das Wissen um die Bibliotheken dahinter (siehe Kapitel 5) zählt zum notwendigen Hintergrundwissen, um zu verstehen, was die anderen Werkzeuge wie APT und `aptitude` überhaupt veranstalten. APT und `aptitude` erleichtern Ihren Alltag als Systembetreuer jedoch deutlich.

8.36.1 Vorbereitungen

Bevor es mit der Installation von Paketen losgeht, prüfen Sie in Schritt 1, ob noch genügend freier Speicherplatz auf Ihrem Linuxsystem verfügbar ist. Damit schließen Sie von vornherein unvollständig im Paketcache zwischengespeicherte und entpackte Pakete (und insbesondere den damit verbundenen Unmut über den sich daraus ergebenden administrativen Zusatzaufwand) aus.

APT ist sehr nett und rechnet Ihnen sogar aus, wieviel zusätzlicher Speicherplatz benötigt wird, wenn Sie das ausgewählte Paket installieren (Schritt 2). Dazu lesen Sie die Nachricht von APT bzw. `aptitude` genau. In der vorletzten Zeile zeigt es Ihnen den

benötigten Speicherplatz für die neuen Pakete an – im nachfolgenden Beispiel für das Paket *kdm* sind es immerhin 36MB. Da im Moment nur der benötigte Speicherplatz von Interesse ist, brechen Sie die Installation ab, indem Sie bei der abschließenden Frage die Taste **n** drücken.

Abgebrochene Installation von kdm mittels APT

```
# apt-get install kdm
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden zusätzlichen Pakete werden installiert:
  kde-wallpapers-default kde-workspace-kgreet-plugins libkworkspace4abi1
Vorgeschlagene Pakete:
  kdepasswd kde-wallpapers
Die folgenden NEUEN Pakete werden installiert:
  kde-wallpapers-default kde-workspace-kgreet-plugins kdm libkworkspace4abi1
0 aktualisiert, 4 neu installiert, 0 zu entfernen und 16 nicht aktualisiert.
Es müssen 33,7 MB an Archiven heruntergeladen werden.
Nach dieser Operation werden 36,3 MB Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren [J/n]? n
Abbruch.
#
```

Abgebrochene Installation von kdm mittels aptitude

```
# aptitude install kdm
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  kde-wallpapers-default{a} kde-workspace-kgreet-plugins{a} kdm libkworkspace4abi1{a}
0 Pakete aktualisiert, 4 zusätzlich installiert, 0 werden entfernt und 16 nicht  ↩
    aktualisiert.
33,7 MB an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 36,3 MB  ↩
    zusätzlich belegt sein.
Möchten Sie fortsetzen? [Y/n/?] n
Abbruch.
#
```

Sind Sie sich unsicher, ob die Installation erfolgreich verlaufen würde, simulieren Sie den Vorgang (Schritt 3). Dazu bietet Ihnen APT die Option `--simulate` und als Alternativen dazu auch `--just-print`, `--dry-run`, `--recon` und `--no-act` an. `aptitude` kennt die Option `-s` bzw. `--simulate` in der Langform. Die nachfolgende Ausgabe zeigt die Simulation anhand von APT und des Pakets *kdm*. Dabei bezeichnen die Zeilen der Ausgabe eine Installation eines Pakets, die mit `Inst` beginnen. Das Schlüsselwort `Conf` besagt, dass das entsprechende Paket konfiguriert wird.

Simulation der Paketinstallation von kdm

```
# apt-get install --simulate kdm
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden zusätzlichen Pakete werden installiert:
  kde-wallpapers-default kde-workspace-kgreet-plugins libkworkspace4abi1
Vorgeschlagene Pakete:
  kdepasswd kde-wallpapers
Die folgenden NEUEN Pakete werden installiert:
  kde-wallpapers-default kde-workspace-kgreet-plugins kdm libkworkspace4abi1
0 aktualisiert, 4 neu installiert, 0 zu entfernen und 16 nicht aktualisiert.
Inst kde-wallpapers-default (4:4.8.4-1 Debian:7.6/stable [all])
Inst kde-workspace-kgreet-plugins (4:4.8.4-6 Debian:7.6/stable [i386])
Inst libkworkspace4abi1 (4:4.8.4-6 Debian:7.6/stable [i386])
Inst kdm (4:4.8.4-6 Debian:7.6/stable [i386])
Conf kde-wallpapers-default (4:4.8.4-1 Debian:7.6/stable [all])
Conf kde-workspace-kgreet-plugins (4:4.8.4-6 Debian:7.6/stable [i386])
Conf libkworkspace4abi1 (4:4.8.4-6 Debian:7.6/stable [i386])
Conf kdm (4:4.8.4-6 Debian:7.6/stable [i386])
```

```
#
```

Als Schritt 4 bringen Sie die Paketliste Ihres Linuxsystems auf den aktuellen Stand. Wie das im Detail vorsichgeht, erfahren Sie in Abschnitt 8.39. Damit stellen Sie sicher, dass Sie mit einer aktuellen Paketliste arbeiten und darüber nur Pakete auswählen, die sich auf dem neuesten Stand befinden. Sie verhindern damit insbesondere, dass Sie veraltete Varianten auf Ihr System einpflegen und reduzieren gleichzeitig den Aufwand bei einer späteren Aktualisierung.

8.36.2 Durchführung

Nachdem alle Vorbereitungen abgeschlossen wurden, folgt nun der Schritt ans Eingemachte – die eigentliche Installation. Für APT lautet der Aufruf `apt-get install Paketname` und für `aptitude` in ähnlicher Art und Weise `aptitude install Paketname`. Beide Werkzeuge verarbeiten nicht nur einzelne Pakete und ganze Paketlisten mit exakten Paketbezeichnungen, sondern auch Paketnamen mit Quantifizierungsoperatoren. Nachfolgender Aufruf zeigt das anhand der Dokumentationspakete zu `aptitude`, deren Namen mit der Zeichenkette *aptitude-doc* beginnen. Damit die Shell, die ihr APT-Kommando ausführt, nicht den Parameter mit dem Quantifizierungsoperator interpretiert und nach Dateien mit dem entsprechenden Namen sucht, schließen Sie das Suchmuster in einfache Anführungszeichen ein.

Aufruf von apt-get install mit dem Quantifizierungsoperator *

```
# apt-get install 'aptitude-doc*'
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Hinweis: »aptitude-doc-cs« wird für regulären Ausdruck »aptitude-doc*« gewählt.
Hinweis: »aptitude-doc-fi« wird für regulären Ausdruck »aptitude-doc*« gewählt.
Hinweis: »aptitude-doc-en« wird für regulären Ausdruck »aptitude-doc*« gewählt.
Hinweis: »aptitude-doc-es« wird für regulären Ausdruck »aptitude-doc*« gewählt.
Hinweis: »aptitude-doc-fr« wird für regulären Ausdruck »aptitude-doc*« gewählt.
Hinweis: »aptitude-doc-ja« wird für regulären Ausdruck »aptitude-doc*« gewählt.
Hinweis: »aptitude-doc-it« wird für regulären Ausdruck »aptitude-doc*« gewählt.
Hinweis: »aptitude-doc« wird für regulären Ausdruck »aptitude-doc*« gewählt.
Die folgenden NEUEN Pakete werden installiert:
  aptitude-doc-cs aptitude-doc-en aptitude-doc-es aptitude-doc-fi aptitude-doc-fr aptitude- ↩
      doc-it
  aptitude-doc-ja
0 aktualisiert, 7 neu installiert, 0 zu entfernen und 16 nicht aktualisiert.
Es müssen 2.337 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 7.642 kB Plattenplatz zusätzlich benutzt.
...
#
```

Wie aus obiger Darstellung ersichtlich wird, durchläuft die Installation eine Reihe von Einzelschritten. Zunächst prüfen APT bzw. `aptitude`, ob das angegebene Paket in der Liste der verfügbaren Pakete überhaupt existiert und ob dieses bereits installiert ist. Danach wird ein Installationskandidat festgelegt und dessen Paketabhängigkeiten werden überprüft. Zu klären ist, ob bereits alle Pakete installiert sind, die von diesem abhängen. Falls das nicht zutrifft, werden diese ebenso mit in die Liste der zu installierenden Pakete aufgenommen. Ist die Liste komplett, wird die Gesamtgröße des zusätzlich belegten Speicherplatzes berechnet.

Bestehen keine Unklarheiten über die zu installierenden Pakete, setzen APT bzw. `aptitude` mit ihrer Arbeit ohne Rückfrage fort. Andernfalls wartet das Programm noch auf Ihr Einverständnis.

Daraufhin werden die Pakete vom Paketmirror bezogen und im Paketcache zwischengespeichert, auf Korrektheit überprüft (siehe Abschnitt 8.35), um die notwendigen Sicherheitsaktualisierungen („Fixes“ oder „Security Updates“) ergänzt und danach temporär ausgepackt. Nun erfolgt die Auswertung und Ausführung der `preinst`-Maintainer-Skripte des Pakets (siehe Abschnitt 4.2) und danach werden die Dateien aus dem jeweiligen Paket an die angegebene Stelle im Dateisystem kopiert. Abschließend erfolgen eine (Nach)Konfiguration (siehe Abschnitt 8.38), sofern das erforderlich ist, und die Ausführung der `postinst`-Maintainer-Skripte des Pakets (siehe Abschnitt 4.2). Ganz am Ende aktualisieren APT bzw. `aptitude` noch die Paketdatenbank und setzen den Status des Pakets auf „vollständig installiert“ (siehe Abschnitt 8.4).

8.36.3 Begutachtung

Nach der Installation gilt es zu überprüfen, ob alles glatt ging. Sie erkennen das an den Rück- und Fehlermeldungen der Programme zur Paketverwaltung.

Im Fehlerfall bieten sowohl APT als auch `aptitude` über die Option `-f` (Langform `--fix-broken`) Rettungshilfe an. Dabei werden fehlende Abhängigkeiten nachinstalliert und defekte Pakete eventuell deinstalliert. Einen Paketnamen müssen Sie im Aufruf nicht angeben, da die Paketzustände ausgewertet werden und darüber entschieden wird, was zu tun ist.

8.36.4 Weitere, nützliche APT-Optionen für den Alltag (Auswahl)

APT kennt eine Reihe von Optionen, die in verschiedenen Situationen im Alltag nützlich sein können. Wir stellen Ihnen hier eine Auswahl davon vor.

`-y` (Langform `--assume-yes` und `--yes`)
: die interaktiven Fragen zur Installation werden automatisch mit „YES“ bzw. „JA“ beantwortet. Die Option ist das Gegenstück zu `--trivial-only`.

`-d` (Langform `--download-only`)
: die Pakete werden nur heruntergeladen, jedoch nicht installiert (siehe Abschnitt 8.31).

`--install-suggests`
: die vorgeschlagenen Pakete werden mitinstalliert.

`--no-install-recommends`
: die empfohlenen Pakete werden nicht automatisch installiert.

`--reinstall`
: das Paket wird erneut installiert (siehe Abschnitt 8.37).

`--trivial-only`
: Gegenstück zur Option `--assume-yes`. Damit werden alle Fragen automatisch mit „NO“ beantwortet und kritische Aktionen bleiben außen vor.

`-V` (Langform `--verbose-versions`)
: in der Ausgabe erscheint die vollständige Versionsangabe des bezogenen Pakets.

8.36.5 Besonderheiten bei `aptitude`

Zwischen APT und `aptitude` bestehen kleine Unterschiede. Dazu zählt auch eine abweichende Vorgehensweise bei Verarbeitung von Paketaktionen. Mit dem Aufruf `aptitude install` installieren Sie nicht nur alle bereits vorgemerkten Pakete, sondern führen alle bereits vorgemerkten Aktionen zur Aktualisierung des Paketbestands durch. Das kann auch die Aktualisierung und Entfernung von Paketen beinhalten. Ausführlicher gehen wir zu dieser Thematik in Kapitel 11 ein.

Bei der Benutzung von Ubuntu besteht eine weitere Besonderheit. Diese betrifft die Voreinstellungen, in der festgelegt ist, dass auch die Empfehlungen eines Pakets mit berücksichtigt werden. Verwenden Sie zum Schalter `install` die Option `-R`, werden nur die direkten Abhängigkeiten mit installiert. Das Verhalten von `aptitude` regeln Sie über den Schlüssel `APT::Install-Recommends` (siehe Abschnitt 10.1).

8.36.6 Erweiterungen ab APT 1.1

Ab der Version 1.1 verfügt APT über interessante Erweiterungen. Diese Version ist vorraussichtlich ab Debian 8 *Jessie* verfügbar. Mit dieser Version können die beiden Aufrufe `apt install` und `apt-get install` nicht nur Paketnamen verarbeiten, sondern auch Pfade zu lokal vorliegenden `deb`-Paketen als Parameter benutzen. Im Gegensatz zu `gdebi` (siehe Abschnitt 6.4.6) besteht hier keine Beschränkung auf nur ein einziges Paket, sondern es gelten die weiter oben benannten Möglichkeiten zur Spezifikation von Paketen. Ein Beispielaufruf sieht wie folgt aus:

Installation eines zweier deb-Pakete ab APT 1.1

```
# apt install /tmp/foo.deb ./bar.deb
...
#
```

8.37 Pakete erneut installieren

Manchmal gehen Dateien kaputt oder werden unbeabsichtigt gelöscht. Sei es bei einer Reparatur des Dateisystems, durch Bit-Dreher auf dem Speichermedium oder durch „Unfälle“ beim Drag-and-Drop. Auf der Kommandozeile passieren mitunter auch Tippfehler, die im Nachhinein nicht mehr reparabel sind.

In anderen Fällen möchten Sie eine Software temporär deinstallieren, um diese später wieder zu installieren. Es ist dabei nicht unüblich, dass bestehende, selbst angepasste Konfigurationsdateien später wiederverwendet werden. Auch das Ersetzen von bereits veränderten Konfigurationsdateien durch ihre Originale aus dem Paket wird oft durch eine Wiederinstallation versucht – nur leider hat dies nur selten den erwünschten Effekt. Warum das nicht immer so funktioniert wie erhofft, erklären wir gleich.

Alle diese Fälle haben gemeinsam, dass ein Paket erneut installiert wird. Dabei unterscheiden wir hier drei verschiedene Ausgangszustände:

- Das Paket wurde mitsamt Konfigurationsdateien entfernt, d.h. vollständig gelöscht (*purged*).
- Das Paket wurde entfernt, die Konfigurationsdateien sind aber noch da.
- Das Paket ist bereits/noch installiert.

Jeder dieser Fälle hat dabei seine Eigenheiten. In den ersten beiden Fällen können Sie das Paket ganz einfach auf den üblichen Wegen installieren. Welche Besonderheiten Sie dabei dennoch beachten sollten, lesen Sie in den folgenden Abschnitten.

Im letztgenannten Fall teilen Ihnen die meisten Programme, die auf APT basieren, mit, dass besagtes Paket bereits installiert ist. Deswegen braucht es dort den expliziten Hinweis, dass das Paket nochmals installiert werden soll. Zum Einsatz kommt dabei meist die Option `--reinstall` (APT) oder das Unterkommando `reinstall` (`aptitude`, `cupt`).

8.37.1 Wiederinstallieren vollständig entfernter Pakete

Wurden Pakete mit der Option `--purge` vollständig entfernt (siehe dazu Abschnitt 8.41), so ist dieser Fall in vielen Fällen trivial und ohne jegliche Besonderheiten.

Die einzige Ausnahme davon bilden Pakete, die Teile ihrer Konfiguration über `debconf` erfragen. Das Kommando `dpkg --purge` entfernt alle Bestandteile außer den in der Debconf-Datenbank gespeicherten Antworten des zu entfernenden Pakets. Dies kann dazu führen, dass Ihnen die `debconf`-Fragen zur Konfiguration des Pakets nicht wieder gestellt werden. Im Endeffekt wird die gleiche Konfiguration wie bei der vorherigen Installation des Pakets generiert.

In diesem Fall hilft Ihnen der Aufruf von `dpkg-reconfigure` mit dem Paketnamen als Parameter. In manchen Fällen reicht dies alleine schon aus, um eine verkorkste Konfiguration wieder hinzubiegen. Mitunter kommen Sie damit um ein gänzliches Entfernen und Wiederinstallieren des Pakets herum. Ausführlich gehen wir dazu unter „Pakete konfigurieren“ in Abschnitt 8.38 ein.

8.37.2 Wiederinstallieren von Paketen mit vorhandenen Konfigurationsdateien

Auch dieser Fall funktioniert meist ganz unspektakulär und so wie Sie es erwarten. Neben den bereits oben erwähnten Stolperstellen kommt jedoch in diesem Fall noch hinzu, dass `dpkg` vorherige Änderungen an den noch vorhandenen Konfigurationsdateien beachtet. Falls sich diese Dateien in der aktuell zu installierenden Paketversion nicht gegenüber denen in der vorher installierten Variante geändert haben, fragt `dpkg` gar nicht nach, ob es diese mit den Varianten aus dem Paket ersetzen soll. Dies gilt auch analog für Konfigurationsdateien, die von Ihnen als Administrator gelöscht wurden. `dpkg` sieht dies als Absicht an, respektiert daher Ihre Entscheidung und installiert die entsprechende Konfigurationsdatei aus dem neuen Paket ebenfalls nicht wieder.

Wurde eine Wiederherstellung erwartet, so muss das Paket vorher von Ihnen mit `purge` (`dpkg --purge`, `apt-get purge` oder `aptitude purge`) entfernt werden. Damit installiert `dpkg` die Konfigurationsdateien aus dem Paket erneut.

8.37.3 Wiederinstallieren bereits installierter Pakete

Der dritte Fall unterscheidet sich von den ersten beiden dahingehend, dass er einerseits erzwungen werden muss und andererseits, dass die gleiche Version wie die bereits vorhandene wieder installiert wird. Andernfalls würde es sich ja um eine Paketaktualisierung handeln.

Diese Vorgehensweise wird meist dann verwendet, wenn – wie zu Beginn dieses Abschnitts erwähnt – eine oder mehrere Dateien eines installierten Pakets kaputt gegangen sind und diese wiederhergestellt werden sollen. Oft ist dies sogar einfacher und schneller, als das Backup zu bemühen.

Ist die entsprechende Paketdatei noch in `/var/cache/apt/archives/` vorhanden, so reicht zum Wiederinstallieren ein simples `dpkg -i` mit der richtigen Datei als Parameter. Da das Paket bereits installiert war, müssen auch alle Abhängigkeiten bereits vorliegen, und es ist nicht notwendig, Abhängigkeiten nochmals aufzulösen.

Muss die Paketdatei neu heruntergeladen werden, so nutzen Sie besser einen der Aufrufe `apt-get install --reinst all`, `aptitude reinstall` oder auch `cupt reinstall`. Zusätzlich benötigen Sie im Aufruf den entsprechenden Paketnamen als Parameter.

Ist das Paket in der aktuell installierten Version jedoch in keinem der dem System bekannten APT-Repositories mehr verfügbar, wird der Vorgang mit einer Fehlermeldung abgebrochen. Desweiteren gelten auch in diesem Fall die gleichen Verhaltensweisen bzgl. geänderter Konfigurationsdateien und `debconf`-Fragen wie in den beiden vorgenannten Fällen.

8.37.4 Typische Stolperfallen bei Wiederinstallieren mehrerer Pakete

Die vermutlich häufigste Stolperfalle beim Wiederinstallieren von Paketen ist, dass Sie das falsche Paket erwischen. Dies passiert im Alltag leider häufiger, als Sie das erwarten würden.

Sind Dateien kaputtgegangen, bei denen Sie sowieso nicht genau wissen, aus welchem Paket diese stammen, so nehmen Sie flink `dpkg -S` zu Hilfe (siehe Abschnitt 8.22). Damit ermitteln damit das dazugehörige Paket im Handumdrehen.

Wissen Sie den Paketnamen jedoch nur ungefähr, wird oft bereits der erste Versuch, das Paket mit dem Namen der Software zu Reinstallieren, ein Fehlschlag. Gerade die größeren Software-Suiten bestehen häufig aus mehreren Paketen und das Paket mit dem Namen der Suite ist meist nur ein Metapaket ohne eigentlichen Inhalt. Zu bedenken ist außerdem, dass eine Wiederinstallation eines bereits installierten Pakets dessen Abhängigkeiten unangetastet lässt.

Ein schönes Beispiel für einen solchen Fall ist die Server-Software namens *Samba*. Haben Sie z.B. die Datei `/etc/pam.d/samba` zerschossen, ist die Versuchung groß, einfach das Paket namens *samba* mit `dpkg --purge` zu deinstallieren und gleich danach wieder zu installieren. Leider wird die Datei danach unverändert sein, da sie nicht zum Paket *samba* gehört, sondern zu dessen Abhängigkeit *samba-common*. Deswegen hilft es Ihnen, im Zweifelsfall doch lieber erst `dpkg -S` zu bemühen und nachzuschauen, in welchem Paket eine Datei wirklich enthalten ist, bevor Sie zu Fluchen anfangen.

8.38 Pakete konfigurieren

Im Normalfall liegen die Debianpakete entweder bereits fertig konfiguriert oder mit einer vorbereiteten Konfiguration auf dem Spiegelserver. Dafür zeichnet der Paketmaintainer verantwortlich.

8.38.1 Bestehende Konfiguration eines Pakets anzeigen

Die bestehende Konfiguration eines Paketes zeigen Sie mit Hilfe des Kommandos `debconf-show` *Paketname* an. `debconf-show` ist Bestandteil des Pakets *dpkg* [Debian-Paket-dpkg] und somit ein essentieller Teil ihrer Installation. Für das Paket *tzdata* [Debian-Paket-tzdata] sieht das Ergebnis wie folgt aus, sofern Sie auf ihrem System als Zeitzone „Europa/Berlin“ eingestellt haben:

Ausgabe der Konfiguration eines Pakets — hier tzdata

```
# debconf-show tzdata
  tzdata/Zones/Pacific:
  tzdata/Zones/Arctic:
```

```
  tzdata/Zones/Africa:
  tzdata/Zones/Asia:
  tzdata/Zones/US:
* tzdata/Zones/Etc: UTC
  tzdata/Zones/Australia:
* tzdata/Zones/Europe: Berlin
* tzdata/Areas: Europe
  tzdata/Zones/Antarctica:
  tzdata/Zones/SystemV:
  tzdata/Zones/Atlantic:
  tzdata/Zones/Indian:
  tzdata/Zones/America:
#
```

In obiger Ausgabe erscheint vor jeder Zeile ein * für die Fragen zur Konfiguration, die Ihnen als Benutzer bereits gestellt wurden. Das kommt beispielsweise dann vor, wenn Sie Debian auf ihrem System einrichten — das Paket *tzdata* ist essentieller Bestandteil des Installationsprozesses.

8.38.2 Konfiguration für alle Pakete auslesen

Die Konfiguration für alle Pakete ist in der Debconf-Datenbank gespeichert. Um diese Konfiguration auszulesen, bedienen Sie sich des Kommandos `debconf-get-selections` aus dem Paket *debconf-utils* [Debian-Paket-debconf-utils]. Dieses Paket gehört nicht zur Basisinstallation und ist daher ggf. noch von Ihnen nachzuinstallieren.

Die Ausgabe des Programms erfolgt zeilenweise auf dem Standardausgabekanal (`stdout`). Sie erhalten zunächst eine Kommentarzeile, die jeweils mit einem # eingeleitet wird. Darauf folgt die Konfigurationsvariable und der Wert, der für die entsprechende Variable derzeit hinterlegt ist. Da die Debconf-Datenbank eine hohe Anzahl Variablen speichert, macht Ihnen ein Pager wie `less` die seitenweise Betrachtung leichter (siehe nachfolgendes Beispiel).

Gespeicherte Konfiguration in der Debconf-Datenbank

```
# debconf-get-selections | less
# Standardwortliste des Systems:
# Choices: american (American English), deutsch (New German), Manuelle Einrichtu
ng von symbolischen Verweisen
dictionaries-common     dictionaries-common/default-wordlist    select  deutsch
(New German)
# Jetzt die Umstellung auf GRUB 2 abschließen?
grub-pc grub-pc/mixed_legacy_and_grub2  boolean true
...
#
```

Das Gegenstück zu `debconf-get-selections` ist `debconf-set-selections` (Paket *debconf-utils* [Debian-Paket-debconf-utils]). Dieses Werkzeug kann die Ausgabe von `debconf-get-selections` direkt verarbeiten und dient Ihnen bspw. zum Einspielen einer vorab gesicherten Paketkonfiguration auf dem gleichen oder einem anderen System. Ausführlicher besprechen wir diesen speziellen Anwendungsfall im Praxisteil unter „Paketkonfiguration sichern“ in Kapitel 35.

8.38.3 Bestehende Konfiguration anwenden

Die Basiseinstellungen eines Pakets funktionieren im Allgemeinen recht gut, haben aber nicht immer einen Bezug zu ihrem konkreten Einsatzfall. Dieser lässt sich selten vollständig vorhersehen. Daher folgt im Rahmen der Paketinstallation ein individuelles Feintuning seitens des Administrators im Rahmen der Voreinstellungen für alle Benutzer oder durch Sie als Benutzer selbst, indem Sie individuelle, lokale Korrekturen vornehmen.

Das Anwenden der vorbereiteten Konfiguration umfasst mehrere Schritte — bspw. das Festlegen von Zugangsdaten (administratives Passwort von MySQL) oder das Starten von benötigten Diensten, bspw. Exim als Mail Transfer Agent (MTA). Stellt `dpkg` dabei fest, dass zwischen der mitgelieferten Konfiguration des neuen Pakets und der bereits bestehenden Konfiguration Unterschiede vorliegen, werden Sie durch das Programm gefragt, welche der beiden Einstellungen zukünftig genutzt werden soll. Das ist sinnvoll und berücksichtigt insbesondere den Fall, dass das Paket bereits bisher auf ihrem System installiert war und von Ihnen händisch auf ihre individuellen Gegebenheiten angepasst wurde. Als Möglichkeiten werden Ihnen hier angeboten:

- die mitgelieferte Konfiguration des neuen Pakets zu benutzen
- die bestehende Konfiguration des installierten Pakets beibehalten
- sich die Unterschiede zwischen beiden anzeigen zu lassen und
- eine Shell zur individuellen Problembehebung zu öffnen.

Bei letzterem bietet sich Ihnen damit die Möglichkeit, bspw. eine Sicherheitskopie der bestehenden Konfiguration anzulegen, bevor Sie diese verändern. Sollte diese neue Konfiguration nicht ihren Erwartungen oder Bedürfnissen entsprechen, können Sie somit jederzeit auf die Sicherheitskopie zurückgreifen. Bei diesem Vorgehen haben Sie als Administrator die Möglichkeit zu sehen, dass überhaupt Unterschiede bestehen und welche Unterschiede das konkret sind. Sie können Ihre derzeitige Konfiguration beibehalten sowie im Zweifelsfall die Konfiguration auf neue (Standard)Werte zurückzusetzen.

In seltenen Fällen geht dieser Prozess schief, d.h. das Paket wurde zwar entpackt, aber nicht konfiguriert. Hintergrund können nicht sauber aufgelöste Paketabhängigkeiten sein, aber auch Fehler im Paket selbst. Ist ein benötigtes Paket nicht konfiguriert und be- bzw. verhindert damit die Installation und Einrichtung weiterer, davon abhängiger Pakete, teilt Ihnen `dpkg` das wie folgt mit:

Fehler in der Konfiguration am Beispiel des Pakets mysql-server

```
...
mysql-server hängt ab von mysql-server-5.5; aber:
  Paket mysql-server-5.5 ist noch nicht konfiguriert.
dpkg: Fehler beim Bearbeiten von mysql-server (--configure):
 Abhängigkeitsprobleme - verbleibt unkonfiguriert
...
Richte docbook-xml ein (4.5-5) ...
update-xmlcatalog: error: entity already registered
dpkg: Fehler beim Bearbeiten von docbook-xml (--configure):
Unterprozess post-installation script gab den Fehlerwert 1 zurück
dpkg: Abhängigkeitsprobleme verhindern Konfiguration von scrollkeeper:
scrollkeeper hängt ab von docbook-xml (>= 4.2-11); aber:
Paket docbook-xml ist noch nicht konfiguriert.
dpkg: Fehler beim Bearbeiten von scrollkeeper (--configure):
 Abhängigkeitsprobleme - lasse es unkonfiguriert
...
```

Hilfreich ist es in diesem Fall, dass Sie zunächst analysieren, welche Pakete unvollständig installiert sind. Dabei hilft Ihnen der Aufruf `dpkg -C`, wobei `dpkg` für letzteres auch die Langform `--audit` kennt.

Auflistung der unvollständig installierten Pakete mittels dpkg

```
# dpkg --audit
Für die folgenden Pakete fehlt die MD5-Prüfsummen-Datei in der Datenbank,
sie müssen erneut installiert werden:
 slib                Portable Scheme library
 vifm                a ncurses based file manager with vi like keybindings
#
```

Als nächsten Schritt schieben Sie die Konfiguration des Pakets an. Für ein einzelnes Paket gelingt Ihnen das mit dem Aufruf `dpkg --configure` *Paketname* und für mehrere Pakete mit `dpkg -a` (Langform `--pending`). Im letztgenannten Fall arbeitet `dpkg` die Liste der unkonfigurierten Pakete nacheinander ab. Die konkrete Reihenfolge der Abarbeitung legt `dpkg` eigenständig fest.

Die Konfiguration eines Pakets geschieht in den folgenden Schritten:

1. Die Konfigurationsdateien („Conffiles“) des Pakets werden entpackt.
2. Die bereits bestehenden Konfigurationsdateien („Conffiles“) für das Paket werden gespeichert. Falls dabei etwas schief geht, können diese wiederhergestellt werden.
3. Stellt das Paket ein Maintainer-Skript namens `postinst` bereit, wird dieses abgearbeitet.

Möchten Sie zu einem späteren Zeitpunkt die Einstellungen zu dem nun installierten und konfigurierten Paket erneut anpassen, benutzen Sie stattdessen das Werkzeug `dpkg-reconfigure`. Damit durchlaufen Sie die Prozedur erneut. Ausführlicher gehen wir dazu in Abschnitt 8.38.4 ein.

8.38.4 Konfiguration mit `dpkg-reconfigure` erneut durchführen

- Aufruf: `dpkg-reconfigure Paket`
- konfiguriert ein bereits installiertes Paket erneut
- verwendet wird dazu `debconf`, welches eine Datenbank mit den Konfigurationseinträgen der Pakete unter `/var/cache/debconf` speichert
- Beispiel:
  - locale-Einstellungen (Sprache, Lokalisierung, Zeichensatz)
  - Einstellung für die Zeitzone (Paket *tzdata* [Debian-Paket-tzdata])

8.39 Pakete aktualisieren

Ein Großteil der verfügbaren Software veraltet häufig in recht kurzer Zeit. Die Entwickler veröffentlichen neue Softwarepakete, die um Fehler bereinigt oder bei denen neue Funktionen ergänzt wurden. Das Debian-Team zur Qualitätssicherung hat daher mehrere Ebenen eingeführt, um einerseits mit der mitunter recht dynamischen Entwicklung der Software schrittzuhalten und andererseits sicherzustellen, dass brandneue Software möglichst keine andere, bereits bestehende und funktionierende Software in Mitleidenschaft zieht.

Dazu gehören die verschiedenen Veröffentlichungen wie *unstable*, *testing* und *stable* (siehe Abschnitt 2.10) sowie der damit festgelegte Zyklus, unter welchen Kriterien von einer Veröffentlichung zur nächsten diffundieren. Mitunter ist das ein recht langer Zeitraum. Desweiteren zählen die verschiedenen Mechanismen dazu, wie Softwarepakete aktualisiert werden. Die dabei verwendeten Begriffe „Update“ und „Upgrade“ sorgen regelmäßig für Verwirrung.

Im Allgemeinen beschreibt ein *Update* eine generelle Aktualisierung und Fehlerbereinigung eines Softwarepakets ohne Änderung der Schnittstelle und unter Beibehaltung des bereits bestehenden Funktionsumfangs. Ein *Upgrade* bezeichnet hingegen eine Aktualisierung eines Softwarepakets zugunsten einer Funktionserweiterung oder Erneuerung, was auch mit einer Veränderung der Schnittstelle einhergehen kann.

APT und `aptitude` kennen die beiden Unterkommandos `update` und `upgrade` in verschiedenen Schweregraden und verknüpfen damit wechselnde Funktionalitäten. Das Team um APT und `aptitude` hat die Bedeutung der Unterkommandos im Laufe der Entwicklung verändert und zudem auch neue Schlüsselworte hinzugefügt. Das zielt darauf ab, die Unterscheidung der Aktionen und deren Benutzung im Alltag zu vereinfachen. Erschwert wird das dadurch, dass sich bestehende Gewohnheiten i.d.R. nur Schritt für Schritt ändern und daher ihre Zeit brauchen, um sich durchzusetzen.

Die Paketverwaltung mittels `apt-get`, `apt` (ab APT 1.0) und `aptitude` kennt jeweils drei Unterkommandos zur Aktualisierung:

- `update`, verwendbar bei `apt-get`, `apt` und `aptitude`,
- `upgrade`, bei `apt` und `aptitude` genannt `safe-upgrade`, sowie
- `dist-upgrade`, bei `apt` und `aptitude` genannt `full-upgrade`.

Jedes Unterkommando beinhaltet eine spezifische Aktualisierung und wirkt sich entweder nur auf die Paketlisten (`update`) oder auf die Pakete selbst aus (`upgrade`, `dist-upgrade`, `safe-upgrade` und `full-upgrade`).

8.39.1 `update`

Das Unterkommando `update` steht in identischer Form und Bedeutung bei den drei Kommandos `apt-get`, `apt` und `aptitude` zur Verfügung. Es dient dabei nur zur Aktualisierung der Paketlisten, die in der Datei `/etc/apt/sources.list` hinterlegt sind. Es beinhaltet jedoch nicht die Erneuerung der Pakete selbst. Auf die Benutzung dieses Unterkommandos gehen wir ausführlich unter Liste der verfügbaren Pakete aktualisieren in Abschnitt 3.13 ein.

8.39.2 `upgrade` und `safe-upgrade`

Dem Unterkommando `upgrade` von `apt-get` entspricht `safe-upgrade` bei `apt` und `aptitude`. Sie aktualisieren damit alle installierten Pakete auf die neueste, verfügbare Version. Dabei werden keine potentiell gefährlichen Aktionen ausgeführt. Was dies genau heißt, unterscheidet sich dezent bei den drei Werkzeugen:

- `apt-get upgrade` ist am konservativsten und installiert weder neue Pakete, noch entfernt es ggf. nicht mehr benötigte Pakete. Dies kann gelegentlich dazu führen, dass nicht alle Sicherheitsaktualisierungen eingespielt werden, wenn diese beispielsweise zum Beheben eines Sicherheitsproblemes zusätzliche Pakete nachsichziehen. Eine solche Situation trat 2008 auf, als eine Sicherheitsaktualisierung für das Paket *openssh-server* eine zusätzliche Abhängigkeit vom Paket *openssh-blacklist* hatte. Letzteres beinhaltet eine schwarze Liste von öffentlich bekannten privaten SSH-Schlüsseln.
- `apt safe-upgrade` läßt hingegen das Installieren neuer Pakete zu. Ähnlich wie `apt-get` entfernt es dabei keine Pakete.
- `aptitude safe-upgrade` geht im Gegensatz zu `apt-get` und `apt` noch einen Schritt weiter und erlaubt auch, dass Pakete entfernt werden. Das betrifft allerdings ausschließlich solche Pakete, die die Markierung „automatisch installiert“ tragen (siehe Paketflags in Abschnitt 2.15). Über die Option `--no-new-installs` sorgen Sie dafür, dass auch `aptitude` beim Aktualisieren nur die Pakete erneuert, die keine weiteren, zusätzlichen Pakete nachsichziehen.

Eine Paketversion wird nicht erneuert und auf dem aktuellen Stand belassen, wenn eine Paketaktualisierung einen weiteren Abhängigkeitskonflikt hervorruft. Das betrifft nur den Fall, wenn ein Paket entfernt werden soll. `aptitude` berücksichtigt dabei nur Pakete, die nicht automatisch über Abhängigkeiten installiert wurden.

Überprüfung der Aktualisierung
Aufgrund der eingebauten Rückhaltemechanismen für potentielle Paketentfernungen werden diese Unterkommandos gerne für Sicherheitsaktualisierungen verwendet. Bitte überprüfen Sie nach deren Ausführung, ob auch alle Aktualisierungen tatsächlich eingespielt wurden. Sollte das nicht der Fall sein, schauen Sie nach, welche Pakete noch ausstehen und welche aufgetretenen Konflikte deren Aktualisierung verhindert haben.

Übersicht zu den aktualisierbaren Paketen erhalten
Welche Pakete aktualisiert werden können, teilen Ihnen APT und `aptitude` mit. Ausführlicher gehen wir darauf unter Abschnitt 8.12 ein.

Sichtbar wird die Änderung auch im Paketnamen. Debian handhabt es so, dass bei Sicherheitsaktualisierungen (genannt *security updates*) dem Paketnamen die Zeichenkette `~deb7u1` für die erste Fehlerbereinigung für Debian 7 *Wheezy* angefügt wird. Die zweite Fehlerbereinigung erhält dann die Zeichenkette `~deb7u2` (siehe dazu auch Abschnitt 2.11). Bei neuen Versionen mit Funktionserweiterungen wird die Versionsnummer des Pakets erhöht.

8.39.3 `dist-upgrade` und `full-upgrade`

Was bei `apt-get` das Unterkommando `dist-upgrade` ist, heißt bei `apt` und `aptitude` hingegen `full-upgrade`. Beide Unterkommandos sind ähnlich zu `upgrade` und `safe-upgrade`.

Sie kommen in zwei Situationen zum Einsatz. Fall eins umfasst das Einspielen von Sicherheitsaktualisierungen, sodass auch neue Abhängigkeiten oder Paketkonflikte Beachtung finden, ohne dass dabei auf die Aktualisierung verzichtet wird. Fall zwei ist der Wechsel von einer Veröffentlichung einer Distribution zur nachfolgenden, so bspw. von *stable* nach *unstable* oder von Debian 7 *Wheezy* nach Debian 8 *Jessie* (siehe auch Distribution aktualisieren in Abschnitt 8.45).

Die bisherigen Veröffentlichungen von APT und `aptitude` suggerierten insbesondere bei dem Begriff `dist-upgrade` inkorrekterweise primär eine Aktualisierung der genutzten Veröffentlichung. Deshalb wurde diese Funktionalität zunächst bei `aptitude` und später auch bei APT von `dist-upgrade` in `full-upgrade` umbenannt. Damit soll klargestellt werden, dass dieses Unterkommando nicht nur zum Wechsel von einer Veröffentlichung zur nächsten (vulgo „Distributions-Upgrade“) anwendbar ist.

In der Funktionalität bestehen kleine Unterschiede:

- mit beiden Unterkommandos werden auch stets neue Pakete installiert, um die Paketabhängigkeiten zu erfüllen. Bei `apt-get` werden gegebenenfalls auch Pakete wieder entfernt, falls ein Paketkonflikt dies erforderlich macht.
- In der Standardeinstellung von `aptitude` entfernt der Aufruf von `aptitude full-upgrade` nicht mehr gebrauchte, automatisch installierte Pakete. Dieses Verhalten können Sie in der Konfiguration von `aptitude` über das Element `Aptitude::Delete-Unused` abschalten.

8.39.4 Empfohlene Schrittfolge zur Aktualisierung von Paketen

Um Ihnen die Aktualisierung Ihrer Softwarezusammenstellung zu vereinfachen, haben wir nachfolgend eine Schrittfolge zusammengestellt, die Ihnen als Orientierung dienen kann. Sind Sie auf der **Kommandozeile** unterwegs, hilft Ihnen diese Abfolge bei den Werkzeugen `apt-get`, `apt` und `aptitude` weiter:

1. Zunächst bringen Sie mittels `apt-get update`, `apt update` oder `aptitude update` die Paketlisten auf den neuesten Stand.
2. Nun aktualisieren Sie mittels `apt-get upgrade`, `apt upgrade` oder `aptitude safe-upgrade` alle Pakete, die keine potentiell gefährlichen Paketoperationen zur Folge haben könnten.
3. Als letzten Schritt führen Sie mit `apt-get dist-upgrade`, `apt full-upgrade` oder `aptitude full-upgrade` eine Erneuerung der Pakete durch, die bisher nicht erneuert wurden. Prüfen Sie bei der Frage "Y/n?" genau die vorgeschlagenen Paketoperationen.

Für das interaktive Arbeiten in der Text-Modus-Oberfläche von `aptitude` ist folgende Reihenfolge sinnvoll:

1. Starten Sie zunächst `aptitude` mit der Option `-u`. Damit aktualisieren Sie zu Beginn die Paketlisten.
2. Mit **[** öffnen Sie die Äste „Aktualisierbare Pakete“ und „Sicherheitsaktualisierungen“, um zu sehen, welche Pakete zur Aktualisierung anstehen.
3. Mit **U** merken Sie alle aktualisierbaren Pakete vor.
4. Eventuelle Konflikte lösen Sie, indem Sie z.B. den ersten Lösungsvorschlag mit **!** akzeptieren.
5. Mit **g** sehen Sie die Vorschau der anstehenden Aktionen an.
6. Drücken Sie nochmals **g**, um die vorbereiteten Aktionen auszuführen.

8.39.5 Aktualisierung mit Synaptic

Über die graphische Oberfläche von Synaptic (siehe Abschnitt 6.4.1) können Sie ebenfalls einzelne oder mehrere Pakete aktualisieren. Welche Aktualisierungen dabei berücksichtigt werden, legen Sie über die Einstellungen des Programms fest. Zu Auswahl stehen hier die Sicherheitsaktualisierungen und neue Paketversionen. Synaptic unterscheidet dabei nicht wie APT, `apt` und `aptitude` zwischen den verschiedenen Aktualisierungsstufen.

Folgende Schritte führen zu neuen Paketen über die graphische Oberfläche:

1. Wählen Sie als erstes den Knopf Status → Installiert (aktualisierbar) aus.
2. Danach selektieren Sie das gewünschte Paket aus der Liste.
3. Über den Menüeintrag Paket → Zum Aktualisieren vormerken fügen Sie dieses zu ihrer Vorauswahl hinzu.
4. Über den Menüpunkt Bearbeiten → Vorgemerkte Änderungen anwenden lösen Sie die Aktualisierung aus.

Ein Distributionswechsel ist nur über vorherige Änderung der Paketquellen möglich. Dabei ergänzen Sie zunächst eine weitere Paketquelle und beziehen danach die Aktualisierung (`update`).

8.40 Pakete downgraden

Allgemein gesprochen, steht der Begriff *Downgrade* für einen Niedergang, eine Abwertung oder einen Rückschritt. Bezogen auf die Verwaltung von Softwarepaketen umfasst es das Einspielen oder Zurückgehen zu einer vorherigen Paketversion. Es stellt damit das Gegenstück zu einer Aktualisierung mittels `apt-get upgrade` dar (siehe dazu Abschnitt 8.39).

Ein Downgrade ist in Betracht zu ziehen, wenn die derzeit installierte Version eines Softwarepakets nicht das leistet, was sie verspricht, bspw. dabei Fehler auftreten oder Inkompatibilitäten mit anderen Softwarepaketen deren Benutzung unmöglich machen. Häufig fallen in letztere Kategorie geänderte Schnittstellen, die noch nicht auf allen nachfolgenden Ebenen konsequent umgesetzt wurden.

Ein Downgrade wird vom Release-Team von Debian GNU/Linux offiziell nicht unterstützt. Alle Mechanismen zur Paketverwaltung und Aktualisierung sind auf das Einspielen einer neueren Version ausgerichtet. Daher verfügt auch keines der hier im Buch vorgestellten Werkzeuge zur Paketverwaltung bislang über einen spezifischen Schalter, um ein Downgrade explizit anzustoßen. Es bleibt daher nur ein Vorgehen über andere Wege, die jedoch auf den bereits zuvor beschriebenen Mechanismen aufsetzen.

8.40.1 Hintergrund und Fragen zum Downgrade

Das Einspielen einer neueren Version ist vom prinzipiellen Ablauf her nicht anders als die Aktualisierung — es laufen die gleichen Mechanismen ab und es kommen die gleichen Werkzeuge zum Einsatz. Der Unterschied ist jedoch die Komplexität, die hier deutlich höher ist.

Vergleichbar ist der Vorgang wie das Bewegen entgegen der Fahrtrichtung in einer Einbahnstraße — es geht so lange gut, wie Ihnen keiner entgegenkommt. Schwierigkeiten können Ihnen nämlich die Maintainerskripte (siehe „Binärpakete“ in Abschnitt 4.2.3) bereiten, die das Downgrade im Normalfall nicht unterstützen. Eventuell ist der Mechanismus, der sie aufruft, auch nicht darauf vorbereitet. Kritisch sind insbesondere die Fälle, wo eine konzeptuelle Änderung im Paket in der Rückrichtung nicht umgebaut werden kann (siehe dazu bspw. den Debian-Bug 764503 [apt-get-update-bug-764503]).

Ein Downgrade ist mit einer Aktualisierung gleichzusetzen. Hierbei benennen Sie jedoch explizit eine ältere Paketversion, die Sie entweder über einen Parameter — „target release“ oder Versionsnummer — beim Aufruf von `dpkg` bzw. `apt-get` angeben oder in der Textoberfläche von Aptitude auswählen. Berechnet die Paketverwaltung nun die Abhängigkeiten zu den übrigen Paketen, kann am Ende dieses Vorgangs auch eine großflächige Änderung am Restbestand der Pakete stehen. Dieser Fall ist nicht ungewöhnlich, denn er kann ebenso bei einer Aktualisierung vorkommen. Die Wahrscheinlichkeit, daß die Änderungen erheblich sind, ist sehr groß.

Wie oben schon benannt, sind diese Änderungen nicht immer rückwärtskompatibel und lösen Verwicklungen aus (Aktualisierungen sind eigentlich bereits hinreichend komplex). Wir empfehlen Ihnen daher, ein Downgrade nur bei dem tatsächlichen Bedarf dafür durchzuführen. Prüfen Sie bitte vorher, ob das Mischen von Veröffentlichungen mittels `apt-pinning` (siehe Kapitel 20) oder das Übersetzen des Pakets aus den Quellen und das nachfolgende Einspielen des eigenen Binärpakets risikoärmer ist.

8.40.2 Ablauf und Durchführung

8.40.2.1 Bestehende Paketversionen klären

Als Schritt eins bringen Sie in Erfahrung, welche Paketversionen überhaupt installiert und darüberhinaus aus dem Repository ihrer genutzten Veröffentlichung verfügbar sind. Dabei helfen ihnen bspw. die Werkzeuge `apt-cache`, `aptitude`, `rmadison` und `apt-show-versions` weiter (siehe „Paketstatus erfragen“ in Abschnitt 8.4.4, „Verfügbare Paketversionen ermitteln“ in Abschnitt 8.18.2 und „Aus welchem Repo kommen die Pakete“ in Abschnitt 8.18). Im Aufruf benötigen alle Programme zusätzlich den Namen des gewünschten Pakets und listen in der Ausgabe die letzte Version auf, ggf. noch spezifiziert für die jeweilige Veröffentlichung. Die nachfolgende Ausgabe nutzt `apt-show-versions` und zeigt das anhand des Paketes `openvpn` aus der stabilen Veröffentlichung.

Auflistung der verfügbaren Versionen zum Paket openvpn

```
$ apt-show-versions openvpn
openvpn:amd64/jessie 2.3.4-5+deb8u1 uptodate
$
```

Die oben benannten Werkzeuge können Ihnen jedoch nicht darstellen, welche vorherigen Versionen eines Pakets existieren und noch verfügbar sind. Aus obiger Ausgabe von `apt-show-versions` wird nur ersichtlich, daß derzeit die Version `2.3.4-5+deb8u1` installiert ist und es sich dabei um das derzeit aktuellste Paket handelt. Das Suffix `deb8u1` deutet auf eine (Sicherheits-)Aktualisierung der Vorgängerversion `2.3.4-5` hin.

Um diese Version aufzuspüren, kann ein Blick in den Paketcache bereits zum Erfolg führen:

Recherche im Paketcache

```
$ ls /var/cache/apt/archives/openvpn*
/var/cache/apt/archives/openvpn_2.3.4-5_amd64.deb
/var/cache/apt/archives/openvpn_2.3.4-5+deb8u1_amd64.deb
$
```

Sie sehen, dass das Paket mit der Version `2.3.4-5` noch lokal herumliegt. Dieses Paket können Sie nachfolgend benutzen.

Sollte obiger Schritt jedoch nicht (mehr) von Erfolg gekrönt sein — weil Sie bspw. den Paketcache aufgeräumt haben — benötigen Sie Plan B. Dieser beinhaltet eine Recherche im Paketarchiv unter *Debian Snapshots* [Debian-Snapshots] (siehe Abbildung 8.20).

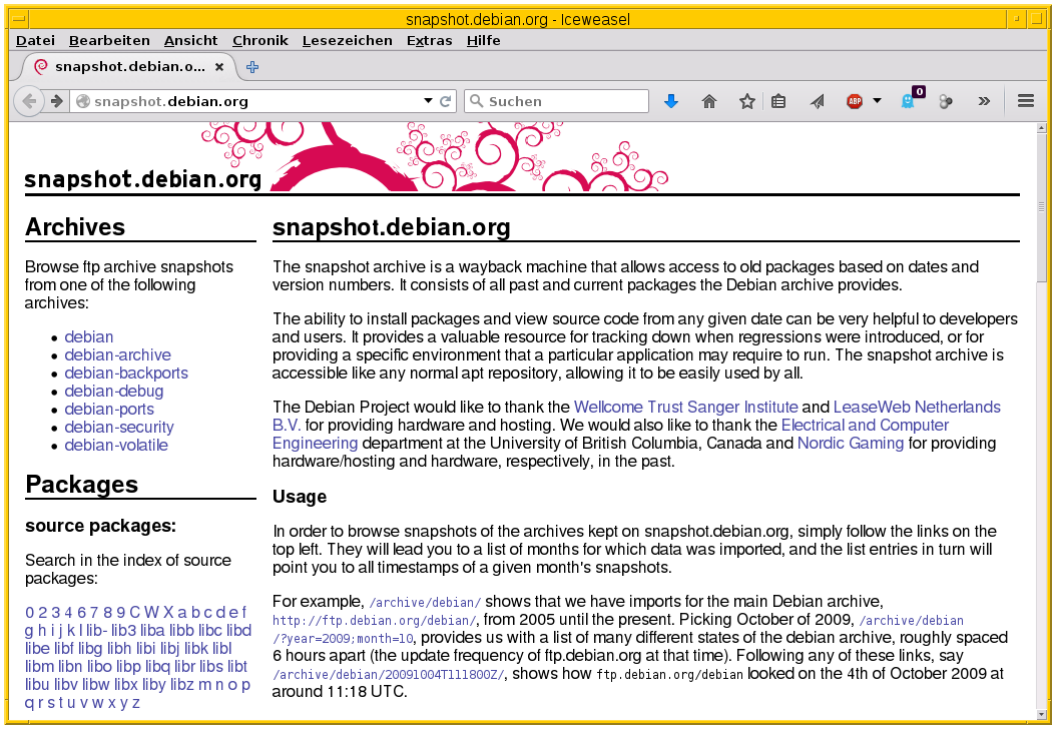

Abbildung 8.20: Das Debian-Paketarchiv

Dieses Archiv beinhaltet den Zugriff auf alle Varianten eines Pakets, welche jemals Bestandteil einer Veröffentlichung von Debian waren. Über diese Webseite stöbern Sie veröffentlichungsbezogen oder anhand des Paketnamens für das Quell- bzw. Binärpaket. Abbildung 8.21 zeigt das Suchergebnis für das Paket ‚openvpn`. Mit einem Klick auf die gesuchte Version aus der Liste beziehen das benötigte Paket aus dem Archiv und speichern es im Paketcache unter `/var/cache/apt/archives‘.

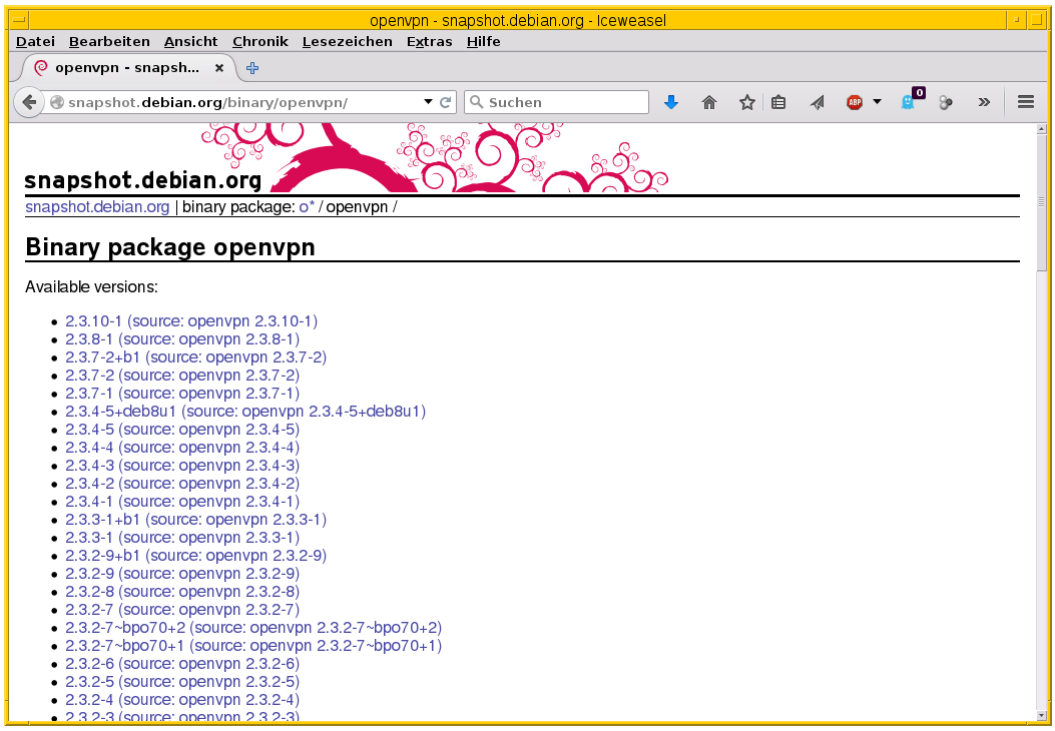

Abbildung 8.21: Suchergebnis nach dem Paket `openvpn` im Paketarchiv

8.40.2.2 Paket austauschen

Im sich nun anschließenden Schritt zwei ersetzen Sie das aktuelle Paket durch dessen Vorgänger. Dieser Schritt ist unkompliziert, sofern keine größeren Paketabhängigkeiten bestehen und repariert werden müssen. Im vorliegenden Fall genügt dazu folgendes:

1. Entfernen des derzeit installierten `openvpn`-Pakets mittels `apt-get remove openvpn`
2. Einspielen des älteren `openvpn`-Pakets mittels `dpkg -ihv /var/cache/apt/archives/openvpn_2.3.4-5_amd64.deb`.

Bei dieser Vorgehensweise bleiben alle Konfigurationsdateien unverändert erhalten.

8.40.2.3 Paket über die Angabe der Versionsnummer austauschen

- über die explizite Angabe der Versionsnummer des Pakets:

```
apt-get install <package-name>=<package-version-number>
```

8.40.2.4 Paket über die Angabe der Veröffentlichung austauschen

- über die explizite Angabe der Veröffentlichung:

```
apt-get -t=<target release> install <package-name>
```

8.41 Pakete deinstallieren

Weitaus anspruchsvoller als die Installation eines Pakets ist hingegen deren rückstandsfreie und saubere Entfernung. Dazu zählt nicht nur das Löschen der Dateien des eigentlichen Pakets, sondern auch das Aufräumen und Entsorgen der dazugehörigen Konfigurationsdateien aus Ihrem Linuxsystem.

Wir unterscheiden daher an dieser Stelle vier Fälle. Fall 1 ist das einfache Deinstallieren eines Pakets, Fall 2 die Recherche, Fall 3 das Entfernen von noch verbliebenen Konfigurationsdateien bereits deinstallierter Pakete sowie als Fall 4 das vollständige Entsorgen von Pakets samt dazugehöriger Konfigurationsdateien in einem einzigen Schritt.

8.41.1 Fall 1: Paket einfach löschen

Dazu dienen die beiden Kommandos `apt-get remove` *Paketname* und `aptitude remove` *Paketname*. Beide Aufrufe entfernen das Paket und ggf. auch alle weiteren Pakete, die davon abhängen. Dabei werden jedoch nur die Daten und die ausführbaren Dateien gelöscht – die dazugehörigen Konfigurationsdateien bleiben unversehrt. Das Vorgehen entspricht dem Aufruf `dpkg -r` *Paketname* in der richtigen Reihenfolge der Paketabhängigkeiten. Der nachfolgende Aufruf zeigt das Vorgehen anhand des Pakets *cssed* für `apt-get`.

Löschen eines Pakets cssed mittels APT

```
# apt-get remove cssed
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Das folgende Paket wurde automatisch installiert und wird nicht mehr benötigt:
  gcr
Verwenden Sie »apt-get autoremove«, um es zu entfernen.
Die folgenden Pakete werden ENTFERNT:
  cssed
0 aktualisiert, 0 neu installiert, 1 zu entfernen und 16 nicht aktualisiert.
Nach dieser Operation werden 2.052 kB Plattenplatz freigegeben.
Möchten Sie fortfahren [J/n]? J
(Lese Datenbank ... 304082 Dateien und Verzeichnisse sind derzeit installiert.)
Entfernen von cssed ...
Trigger für man-db werden verarbeitet ...
Trigger für menu werden verarbeitet ...
Trigger für gnome-menus werden verarbeitet ...
Trigger für desktop-file-utils werden verarbeitet ...
#
```

Ein Knackpunkt stellt die Berücksichtigung der jeweiligen Paketabhängigkeiten dar. Dabei treten mehrere Möglichkeiten auf – bestimmte, zur Löschung vorgesehene Pakete werden von anderer Software noch benötigt, Ersetzungen sind erforderlich oder es entstehen Waisen (*Orphans*). Bei Möglichkeit eins dürfen die Pakete, die von anderer Software noch benötigt werden, nicht gelöscht werden – die andere installierte Software soll ja trotzdem weiterhin funktionieren. Bei systemrelevanten Werkzeugen in essentiellen Paketen erhalten Sie daher eine zusätzliche, deutliche Warnung (siehe nachfolgendes Beispiel sowie Paketprioritäten und essentielle Pakete in Abschnitt 2.13).

Ausgabe einer deutlichen Warnung vor dem Löschen des essentiellen Pakets base-files

```
# apt-get remove base-files
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut
Statusinformationen werden eingelesen... Fertig
Die folgenden Pakete werden ENTFERNT:
  base-files bash bash-completion bsd-mailx build-essential clisp common-lisp-controller ↩
      debhelper
  dpkg-dev foomatic-db-engine foomatic-filters grsync grub grub-common grub-pc gt5 libnss- ↩
      mdns rsync
  virtualbox-ose-guest-source virtualbox-ose-source xindy
WARNUNG: Die folgenden essentiellen Pakete werden entfernt.
```

```
Dies sollte NICHT geschehen, außer Sie wissen genau, was Sie tun!
  base-files bash
0 aktualisiert, 0 neu installiert, 21 zu entfernen und 138 nicht aktualisiert.
Nach dieser Operation werden 32,4 MB Plattenplatz freigegeben.
Sie sind im Begriff, etwas potentiell Schädliches zu tun.
Zum Fortfahren geben Sie bitte »Ja, tue was ich sage!« ein.
Abbruch.
#
```

Möglichkeit zwei ist die Ersetzung durch ein alternatives Paket. Das gelingt dann automatisch, wenn in der Paketbeschreibung als Paketabhängigkeit entweder mehrere Einzelpakete oder ein einzelnes, virtuelles Paket benannt wurden. Aus dieser Liste wird dann eines ausgewählt, um die Paketabhängigkeit zu erfüllen. Bestes Beispiel ist das virtuelle Paket *pdf-viewer*, welches beispielsweise auf *epdfview*, *evince*, *okular*, *xpdf* und *zathura* verweist.

Ersetzung durch ein alternatives Paket

```
ToDo: Beispiel mit Ersetzung durch alternatives Paket
```

Bei der Möglichkeit drei entstehen Waisen – Pakete, die keine Abhängigkeiten mehr zu anderen Paketen mehr aufweisen. Unter Umgang mit Waisen in Abschnitt 8.42 beleuchten wir dieses Thema näher.

8.41.2 Fall 2: Suche von Konfigurationsdateien bereits deinstallierter Pakete

Um das zu tun, bedarf es zunächst der Lokalisierung der Pakete, welche zwar gelöscht wurden, aber noch als konfiguriert gelten. Dabei geht es nur um die Konfigurationsdateien (*conf files*), die sich unter dem Verzeichnis `/etc` befinden. Dateien in ihrem Homeverzeichnis bleiben unberührt.

Die passenden Werkzeuge sind dafür die Kombination aus `dpkg` mit `grep` sowie `aptitude`. APT hat u.E. bislang keinen entsprechenden Schalter dafür.

`dpkg` rufen Sie dazu zunächst mit der Option `-l` auf (siehe Abschnitt 8.5) und schicken dessen Ausgabe an das Kommando `grep` weiter. Mit dem regulären Ausdruck `"^rc "` (mit Leerzeichen am Ende) filtern Sie alle Zeilen aus der Ausgabe heraus, die mit den beiden Buchstaben `rc` beginnen und von einem Leerzeichen gefolgt werden. Damit erhalten Sie eine Liste aller verbliebenen Konfigurationsdateien, die `dpkg` einem Paket zuordnen kann.

Suche nach gelöschten, aber noch konfigurierten Paketen mittels dpkg

```
$ dpkg -l | grep "^rc "
rc  akonadi-backend-mysql 1.7.2-3   all   MySQL storage backend for Akonadi
rc  akonadi-server        1.7.2-3   i386  Akonadi PIM storage service
rc  atop                  1.26-2    i386  Monitor for system resources and process activity
rc  audtty                0.1.12-1  i386  ncurses based frontend to audacious
...
$
```

Auch `aptitude` kann in diese Richtung recherchieren. Es kennt zu diesem Zweck zum Schalter `search` die Option `~c` bzw. die Langform `?config-files`. Das Ergebnis umfasst jedoch *alle* konfigurierten Pakete – unabhängig davon, ob diese als gelöscht markiert sind oder nicht.

Suche nach konfigurierten Paketen mittels aptitude

```
$ aptitude search ~c
c   akonadi-backend-mysql   - MySQL-Speicher-Backend für Akonadi
c   akonadi-server          - PIM-Speicherdienst Akonadi
c   atop                    - Überwachung für Systemressourcen und Proze
c   audtty                  - auf ncurses basierende Oberfläche für auda
...
$
```

8.41.3 Fall 3: Löschen von Konfigurationsdateien bereits deinstallierter Pakete

Haben Sie die aus Ihrer Sicht unnützen Konfigurationsdateien eines bereits deinstallierten Pakets ausfindig gemacht und möchten diese endgültig ins digitale Nirwana befördern, sind Ihnen APT und `aptitude` gern dabei behilflich. APT unterstützt Sie mit der Kombination aus dem Kommando `apt-get`, dessen Schalter `remove` und der Option `--purge` *Paketname*. Bei `aptitude` besteht kein separater Schalter für den Schritt – stattdessen genügt hier der Schalter `purge` vollkommen.

Löschen der Konfigurationsdateien mittels APT

```
# apt-get remove --purge cssed
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden Pakete werden ENTFERNT:
  cssed*
0 aktualisiert, 0 neu installiert, 1 zu entfernen und 16 nicht aktualisiert.
Nach dieser Operation werden 0 B Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren [J/n]?
(Lese Datenbank ... 304031 Dateien und Verzeichnisse sind derzeit installiert.)
Entfernen von cssed ...
Löschen der Konfigurationsdateien von cssed ...
Trigger für menu werden verarbeitet ...
#
```

8.41.4 Fall 4: Paket samt Konfigurationsdateien deinstallieren

APT und `aptitude` ermöglichen auch das Deinstallieren eines oder mehrerer Pakete samt zugehöriger Konfigurationsdateien in einem einzigen Schritt. Die Aufrufe entsprechen dem Kommando `dpkg -P` *Paketname* für eine Menge von Paketen in der richtigen Reihenfolge der Paketabhängigkeiten.

Für diese Aktion kombinieren Sie entweder `apt-get` mit dem Schalter `remove` und der Option `--purge` *Paketname*, `aptitude` kennt stattdessen nur den Schalter `purge`.

Löschen des Pakets cssed samt Konfigurationsdateien in einem Schritt

```
# aptitude purge cssed
Die folgenden Pakete werden ENTFERNT:
  cssed{p}
0 Pakete aktualisiert, 0 zusätzlich installiert, 1 werden entfernt und 16 nicht  ↩
    aktualisiert.
0 B an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 2.052 kB frei  ↩
    werden.
Möchten Sie fortsetzen? [Y/n/?]
(Lese Datenbank ... 304082 Dateien und Verzeichnisse sind derzeit installiert.)
Entfernen von cssed ...
Löschen der Konfigurationsdateien von cssed ...
Trigger für man-db werden verarbeitet ...
Trigger für menu werden verarbeitet ...
Trigger für gnome-menus werden verarbeitet ...
Trigger für desktop-file-utils werden verarbeitet ...

#
```

Ein Sonderfall ist das Entfernen aller Pakete für eine bestimmte Architektur. Das tritt auf, wenn Sie bspw. mit dem *Multiarch*-Feature experimentieren (siehe Abschnitt 1.2.3). Um alle Pakete für die Architektur *i386* vollständig zu entfernen, nutzen Sie diesen Aufruf:

Vollständiges Entfernen aller installierten Pakete für die Architektur i386

```
# apt-get remove --purge ".*:i386"
```

8.42 Umgang mit Waisen

Während der Verwendung von APT und `aptitude` werden die Abhängigkeiten der Pakete automatisch aufgelöst und daher auch zusätzlich benötigte Pakete eingespielt. Löschen Sie zu einem späteren Zeitpunkt Pakete, werden i.d.R. alle nicht mehr benötigten Pakete wieder entfernt. Leider gelingt dieser Vorgang nicht immer vollständig und es verbleiben mitunter „Reste“ auf dem System zurück, die ohne Bezug zu anderen Paketen sind. Hängt ein Paket von keinem weiteren mehr ab und bildet kein eigenständiges Programm, wird es daher zu einem Waisen – engl. *orphan*. Häufig betrifft das Pakete aus den Kategorien für Bibliotheken (*libs*), veraltete Bibliotheken (*oldlibs*) und Entwicklungsbibliotheken (*libdevel*).

Der Umgang mit Waisen ist im Allgemeinen recht unproblematisch. Waisen können Sie bedenkenlos löschen, um unnötigerweise belegten Speicherplatz auf ihrem System wieder freizugeben. Wir empfehlen Ihnen, diesen Aufräumvorgang bei der Systempflege als weiteren Schritt mit durchzuführen. Das hält ihr System sauber und befreit es von unnützen Lasten.

Neben den Mechanismen von APT und `aptitude` existieren eine ganze Reihe von weiteren Programmen, um Waisen aufzuspüren und auch zu entfernen. Für die Kommandozeile sind das `debfoster` (siehe Abschnitt 8.42.2) und `deborphan` (siehe Abschnitt 8.42.3). Auf Ncurses basieren Orphaner und Editkeep (siehe Abschnitt 8.42.4) und auf GTK+ das Pendant Gtkorphan (siehe [Gtkorphan]). Darüberhinaus bieten die `aptsh` (siehe Abschnitt 8.42.6) und `wajig` (Abschnitt 8.42.7) entsprechende Möglichkeiten zur Suche, die wir Ihnen nicht ebenfalls vorenthalten möchten.

8.42.1 APT und `aptitude`

Zwischen der Standardkonfiguration von APT und `aptitude` gibt es subtile Unterschiede, die sich über die Zeit herausgebildet haben und die es im Alltag zu beachten gilt. Kurz gefasst, beläßt APT verwaiste Pakete, während `aptitude` diese automatisch entfernt.

APT-Versionen aus der Prä-Lenny-Ära – d.h. vor Debian 5.0 *Lenny* (2009) – nehmen auf die Erzeugung von Waisen kaum Rücksicht. Spätere Veröffentlichungen von APT achten deutlich stärker darauf und weisen den Benutzer darauf hin. Ohne explizite Aufforderung entfernt APT keine Waisen.

Bei `aptitude` ist das ganze Prozedere ein klein wenig anders. `aptitude` räumt hinterher eigenständig auf. Das betrifft jedoch nur automatisch installierte Pakete, von denen wiederum kein manuell installiertes Paket abhängt (keine *reverse dependencies* bestehen).

Über die `aptitude`-Option `aptitude::Delete-Unused` schalten Sie dieses Verhalten zu oder ab – entweder über die Benutzeroberfläche unter Optionen → Einstellungen, oder direkt in der Konfigurationsdatei von `aptitude`.

Abbildung 8.22: Nicht verwendete Pakete automatisch entfernen in `aptitude`

Um den Vorgang der Paketentfernung explizit anzustoßen, verfügen APT und `aptitude` über das Unterkommando `autoremove`. Seitdem `apt-get` ebenfalls dieses Feature besitzt, hat die Bedeutung von `deborphan` und `debfoster` deutlich abgenommen.

Pakete automatisch entfernen mit dem Unterkommando autoremove

```
# apt-get autoremove
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 13 nicht aktualisiert.
#
```

`aptitude` verfügt zudem über die Option `--purge-unused`, die noch einen Schritt weiter geht. Alle Pakete, die `aptitude` mangels Notwendigkeit entfernt, werden inklusive der dazugehörigen Konfigurationsdateien entsorgt. Diese Option können Sie über den Eintrag `aptitude::Purge-Unused` aktivieren.

Verwendung der Option --purge-unused
Diese Option ist sehr mächtig und kombiniert eine ganze Reihe von Einzelschritten. Wir raten Ihnen daher, die Anwendung vorab genau zu prüfen.

8.42.2 debfoster

Das Paket `debfoster` [Debian-Paket-debfoster] ist ein Wrapper für die beiden Werkzeuge `dpkg` und APT. `debfoster` pflegt eine Liste mit den Paketen, die Sie auf ihrem System behalten möchten und auf die Sie Wert legen.

Mit Hilfe dieser Liste findet es Pakete, die automatisch installiert wurden, nur weil andere Pakete davon abhängen. Falls diese Abhängigkeiten nicht mehr bestehen – d.h. ein entsprechendes Paket wurde entfernt – bekommt `debfoster` das mit und fragt Sie, ob Sie das über die Abhängigkeit benannte Paket ebenfalls mit entfernen möchten.

Zu Beginn erstellt `debfoster` auf der Basis Ihrer Rückmeldung eine Liste mit den derzeit installierten Paketen. Diese Liste speichert `debfoster` in der Datei `/var/lib/debfoster/keepers`. Darin vermerkt es, ob Sie das betreffende Paket behalten oder entfernen möchten. Zum Schluss löscht es die Pakete, die in der Liste als „entfernen“ gekennzeichnet sind. Ein Aufruf zur Aktualisierung der Liste ist nach jeder Änderung des Paketbestandes sinnvoll, d.h. einer Installation, Löschung und Aktualisierung eines oder mehrerer Pakete.

Mit dem Kommando `debfoster -qv` erstellen Sie eine initiale Liste. Bei einem Folgeaufruf zeigt es Ihnen die Pakete, die die unerfüllte Abhängigkeiten aufweisen plus möglicherweise nicht mehr benötigte Pakete. `debfoster` warnt bei unerfüllten Abhängigkeiten (*warning*), wenn diese Pakete in der Liste der „zu behaltenden Pakete“ stehen.

Auflistung der unerfüllten Abhängigkeiten mit debfoster

```
# debfoster -qv

warning: package gnome-session-fallback: unsatisfied dependency on notification-daemon 0.7
warning: package gnome-session-fallback: forcing depdency on notification-daemon
warning: package timidity: unsatisfied dependency on libjack-jackd2-0 1.9.5~dfsg-14
warning: package libreoffice-filter-mobiledev: unsatisfied dependency on default-jre
warning: package libreoffice-filter-mobiledev: unsatisfied dependency on gcj-jre
warning: package libreoffice-filter-mobiledev: unsatisfied dependency on java-gcj-compat
...
Paket wird behalten: gdm3
Paket wird behalten: krita
Paket wird behalten: xfce4-goodies
Paket wird behalten: libreoffice
Paket wird behalten: bluetooth
Paket wird behalten: asciidoc
...
#
```

`debfoster` verfügt über eine Reihe von weiteren Optionen. Nachfolgende Liste ist eine Auswahl bzgl. der Thematik „Waisen“, ausführlicher ist die Manpage zum Programm.

`-q` (Langform `--quiet`)
keine Darstellung der Fragen und als Standardantwort *yes*. Sinnvoll zur initialen Erzeugung der Paketliste.

`-f` (Langform `--force`)
keine Darstellung der Fragen und als Standardantwort *no*. Installiert fehlende Pakete nach, wobei die Paketliste maßgeblich ist.

`-v` (Langform `--verbose`)
Statusmitteilung darüber, welche Pakete verschwunden sind, Waisen oder Abhängigkeiten wurden.

`-d` (Langform `--show-depends`)
gebe alle Pakete an, von denen das Paket abhängt. Die Option ist das Gegenstück zur Option `-e` und vergleichbar mit dem Unterkommando `depends` des Programms `apt-cache` (siehe Abschnitt 8.17).

Ausgabe aller Abhängigkeiten mittels debfoster

```
# debfoster -d vim
Paket vim hängt ab von:
  gcc-4.7-base libacl1 libattr1 libc-bin libc6 libc6-i686 libgcc1 libgpm2 libselinux1  ↩
      libtinfo5
  multiarch-support vim-common vim-runtime
#
```

`-e` (Langform `--show-dependents`)
gebe alle Pakete an, die von dem Paket abhängen. Diese Option ist das Gegenstück zur Option `-d` und vergleichbar mit dem Unterkommando `rdepends` des Programms `apt-cache` (siehe Abschnitt 8.17).

Ausgabe aller umgekehrten Abhängigkeiten mit debfoster

```
# debfoster -e apt
Die folgenden 9 Pakete auf der Aufbewahrungsliste verlassen sich auf apt:
  xara-gtk synaptic packagesearch gtkorphan debfoster asciidoc installation-report totem  ↩
      gdm3
Pakete bewahrt durch Standardregeln sich verlassen auf apt.
#
```

`-s` (Langform `--show-orphans`)
auflisten aller Paketwaisen

`-i` (Langform `--ignore-default-rules`)
durch alle Pakete gehen, die explizit installiert wurden

`-a` (Langform `--show-keepers`)
Ausgabe der `debfoster`-Datenbank

Ausgabe der Pakete, die sich debfoster gemerkt hat

```
# debfoster -a
Die folgenden Pakete stehen auf der Aufbewahrungsliste:
  abiword acpi acpi-support anacron apache2-utils apcalc apmd app-install-data apt-doc
  apt-dpkg-ref apt-rdepends aptsh apvlv aqbanking-tools arora ascii asciidoc ash aspell-de  ↩
      at
...
#
```

8.42.3 deborphan

Das Programm `deborphan` aus dem gleichnamigen Debian-Paket [Debian-Paket-deborphan] findet ungenutzte Pakete, die keine weiteren Abhängigkeiten zu anderen Paketen (siehe Abschnitt 8.17) aufweisen. Es gibt Ihnen eine Liste aller gefundenen Pakete aus, die Sie entfernen *sollten*, aber nicht *müssen*. Grundlage für die Liste sind die Paketabhängigkeiten, die `deborphan` über `dpkg` und über die Angaben in der Paketbeschreibung zur Verfügung stehen.

Rufen Sie `deborphan` ohne Optionen auf, beschränkt es sich auf die beiden Paketkategorien *libs* und *oldlibs*, um unbenutzte oder veraltete Bibliotheken zu ermitteln. Das nachfolgende Beispiel zeigt diesen Aufruf beispielhaft.

Ausgabe von deborphan bei der Suche nach verwaisten Paketen

```
$ deborphan
mktemp
liblwres40
libdvd0
libxapian15
libdb4.6
libdb4.5
libevent1
librrd4
libbind9-40
diff
dhcp3-common
$
```

`deborphan` verfügt über eine ganze Reihe nützlicher Optionen. Daraus zeigen wir die Optionen, die uns für die Thematik „Waisen“ relevant erscheinen. Zu weiteren Optionen gibt Ihnen die Manpage des Programms Auskunft.

`-a` (Langform `--all-packages`)
: durchsucht die gesamte Paketdatenbank (siehe Abschnitt 3.14)

`--libdevel`
: durchsucht nicht nur die Paketkategorien *libs* und *oldlibs*, sondern zusätzlich auch die Liste der Entwicklerbibliotheken (*libdevel*)

`-z` (Langform `--show-size`)
: Ausgabe mit Größenangabe des Pakets. Daraus ersehen Sie, wieviel Platz das Paket auf der Festplatte belegt.

`-P` (Langform `--show-priority`)
: Ausgabe zeigt die Priorität des Pakets (siehe Abschnitt 2.13) an; Wert aus *required*, *important*, *standard*, *optional* oder *extra*.

`-s` (Langform `--show-section`)
: zeigt die Paketkategorie (siehe Abschnitt 2.8) an, in dem sich das Paket befindet. Ist die Option standardmäßig aktiviert, können Sie das Verhalten mit der Option `--no-show-section` wieder abschalten.

Auflistung der verwaisten Bibliotheken inkl. Paketkategorie und Größe mittels deborphan

```
$ deborphan -P -z -s
  20 main/oldlibs  mktemp      extra
 132 main/libs     liblwres40  standard
 172 main/libs     libdvd0     optional
 ...
$
```

Kompakte Schreibweise der Optionen
Für den obigen Aufruf existiert eine Kurzschreibweise, in der Sie die Optionen in kompakter Form schreiben können. Der Aufruf `deborphan -Pzs` bewirkt das gleiche wie `deborphan -P -z -s`.

`deborphan` verfügt zudem über einen *Ratemodus*, um Pakete zu finden, die für Sie nicht mehr nützlich sein könnten. Es analysiert dazu den Paketnamen und die Paketbeschreibung. Die Basis bilden die Optionen `--guess-` und `--no-guess-`, die Sie mit entsprechenden Suffixen zur genaueren Eingrenzung kombinieren können. Dazu zählen bspw. `common`, `data`, `dev`, `doc` und `mono`, aber auch `perl`, `pike`, `python` und `ruby` für die entsprechenden Programmiersprachen. Eine ausführliche Auflistung ist in der Manpage dokumentiert.

deborphan errät nicht mehr nützliche Pakete

```
# deborphan --guess-perl | sort
gqview
libchromaprint0
libconsole
libcrypt-rc4-perl
libgraphics-magick-perl
libimage-exiftool-perl
libindicate-gtk3
libpdf-api2-perl
librpcsecgss3
librrd4
libtext-pdf-perl
...
#
```

Mit der Option `--find-config` suchen Sie nach nicht installierten Paketen, von denen noch *Konfigurationsdateien* auf dem System vorliegen. Das impliziert die Option `-a` und durchsucht die gesamte Paketdatenbank. Das nachfolgende Beispiel sortiert zusätzlich die Paketliste alphabetisch aufsteigend und gibt die Ausgabe seitenweise über den Pager `more` auf dem Terminal aus.

Aufspüren nicht mehr benötigter Konfigurationsdateien über die Option --find-config

```
$ deborphan --find-config | sort | more
baobab
bluez-utils
dhcdbd
dpatch
dvipdfmx
gnome-screenshot
--More--
$
```

Für das Paket *gnome-screenshot* aus obiger Ergebnisliste ergibt eine Suche über `dpkg` die nachfolgende Ausgabe. Die Buchstaben `rc` zu Beginn der Zeile mit den Paketdetails zeigen, dass dieses Paket bereits auf dem System installiert war und zwischenzeitlich wieder entfernt wurde (Buchstabe `r` für `removed` in der ersten Spalte). Die Konfigurationsdateien des Programms sind noch verfügbar (Buchstabe `c` für `configured` in der zweiten Spalte).

Aufspüren verbliebener Konfigurationsdateien mittels dpkg

```
$ dpkg -l gnome-screenshot
Gewünscht=Unbekannt/Installieren/R=Entfernen/P=Vollständig Löschen/Halten
| Status=Nicht/Installiert/Config/U=Entpackt/halb konFiguriert/
         Halb installiert/Trigger erWartet/Trigger anhängig
|/ Fehler?=(kein)/R=Neuinstallation notwendig (Status, Fehler: GROSS=schlecht)
||/ Name                Version            Beschreibung
+++-==================-==================-================================
rc  gnome-screenshot    2.30.0-2           screenshot application for GNOME
$
```

Darstellung des Paketstatus
Die ersten beiden Zeichen in der Zeile mit den Paketdetails haben eine besondere Bedeutung und geben den Status des Pakets an. Unter Paketstatus erfragen in Abschnitt 8.4 stellen wir Ihnen alle weiteren Varianten und deren Bedeutung vor.

Um die verbliebenen Konfigurationsdateien eines Pakets auch noch zu entfernen, benutzen Sie üblicherweise das Kommando `apt-get --purge remove Paketname`. Für das oben genannte Paket *gnome-screenshot* heißt der Aufruf `apt-get --purge remove gnome-screenshot`. Weitere Details dazu finden Sie unter Pakete deinstallieren in Abschnitt 8.41.

Eine zusätzliche Möglichkeit bietet die Kombination aus `apt-get` und `deborphan`. Die Angabe `$(deborphan)` bewirkt die Ausführung des Kommandos `deborphan` in einer Subshell und liefert als Rückgabewert alle Pakete, die Waisen sind. Indem Sie das als Parameter an APT übermitteln, sparen Sie einerseits Tipparbeit und können darüberhinaus auf die Rückfragen von APT reagieren.

Kombinieren von APT und deborphan

```
# apt-get --purge remove $(deborphan)
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden Pakete werden ENTFERNT:
  gqview* libchromaprint0* libconsole* libindicate-gtk3* librpcsecgss3*
  librrd4* linux-image-2.6-686* mktemp* pdfjam* qemulator*
  ttf-linux-libertine* virtualbox-ose* virtualbox-ose-dkms*
  virtualbox-ose-guest-source* virtualbox-ose-guest-utils*
  virtualbox-ose-source*
0 aktualisiert, 0 neu installiert, 16 zu entfernen und 8 nicht aktualisiert.
Nach dieser Operation werden 2.517 kB Plattenplatz freigegeben.
Möchten Sie fortfahren [J/n]?
...
#
```

Entsorgen von Waisen

Wenden Sie das Nachfolgende nur an, wenn Sie wissen, was Sie tun, und sich dessen sicher sind. Das Kommando entsorgt kompromisslos alle Waisen und deren Konfigurationsdateien. Die Option `-y` beantwortet alle Nachfragen von `apt-get` automatisch mit „ja“:

Komplexer Aufruf von deborphan

```
# deborphan | xargs apt-get --purge remove -y
```

8.42.4 Orphaner und Editkeep

`orphaner` und `editkeep` sind beides Benutzeroberflächen für `deborphan` (siehe Abschnitt 8.42.3) und Bestandteil des gleichnamigen Pakets [Debian-Paket-deborphan]. Ersteres findet und entfernt verwaiste Pakete, das Zweite hilft Ihnen bei der Pflege und Zusammenstellung der Liste der Pakete, die *nie* von `deborphan` entfernt werden.

`orphaner` und `editkeep` sind beides Shellskripte und rufen nach der Auswahl direkt `apt-get` bzw. `deborphan` mit den passenden Optionen auf. Diese beiden Programme verfügen über ein recht ähnliches Ncurses-Interface. Dargestellt werden zwei Spalten – links der Paketname und rechts der der Distributionsbereich (siehe Abschnitt 2.9) und die Kategorie (siehe Abschnitt 2.8), in die das Paket eingeordnet ist. Über die Buchstabentasten bewegen Sie den Auswahlbalken zum entsprechenden Menüpunkt. Mit der Leertaste ergänzen bzw. entfernen Sie das betreffende Paket von der Auswahl. Mit der Eingabetaste legt das Programm los.

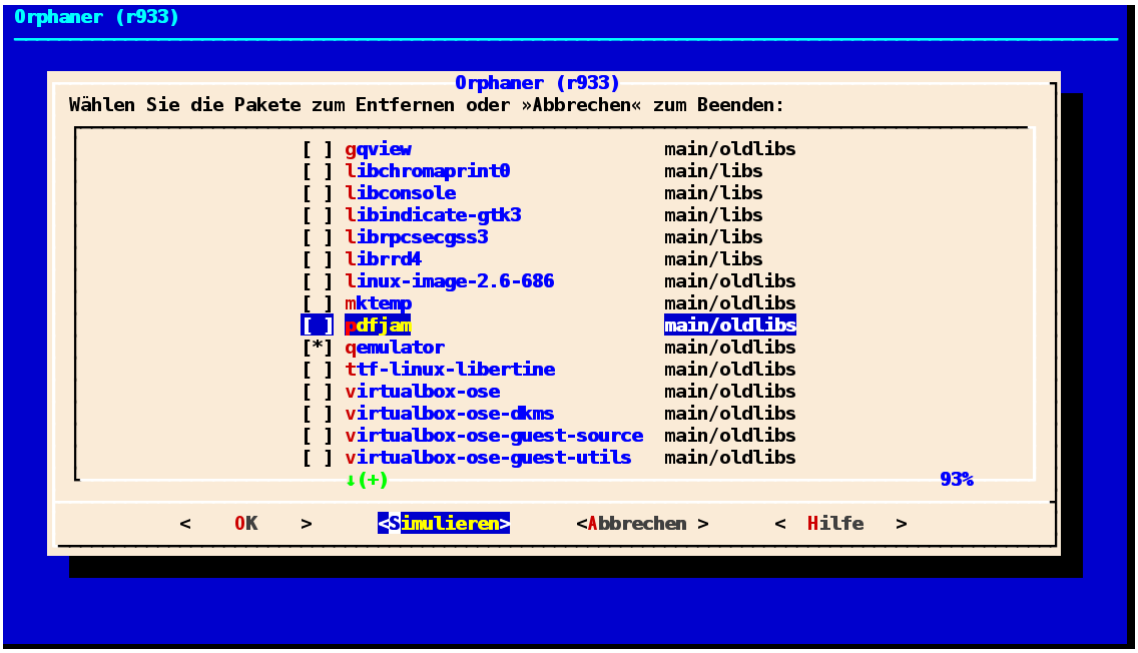

Abbildung 8.23: `orphaner` bei der Arbeit

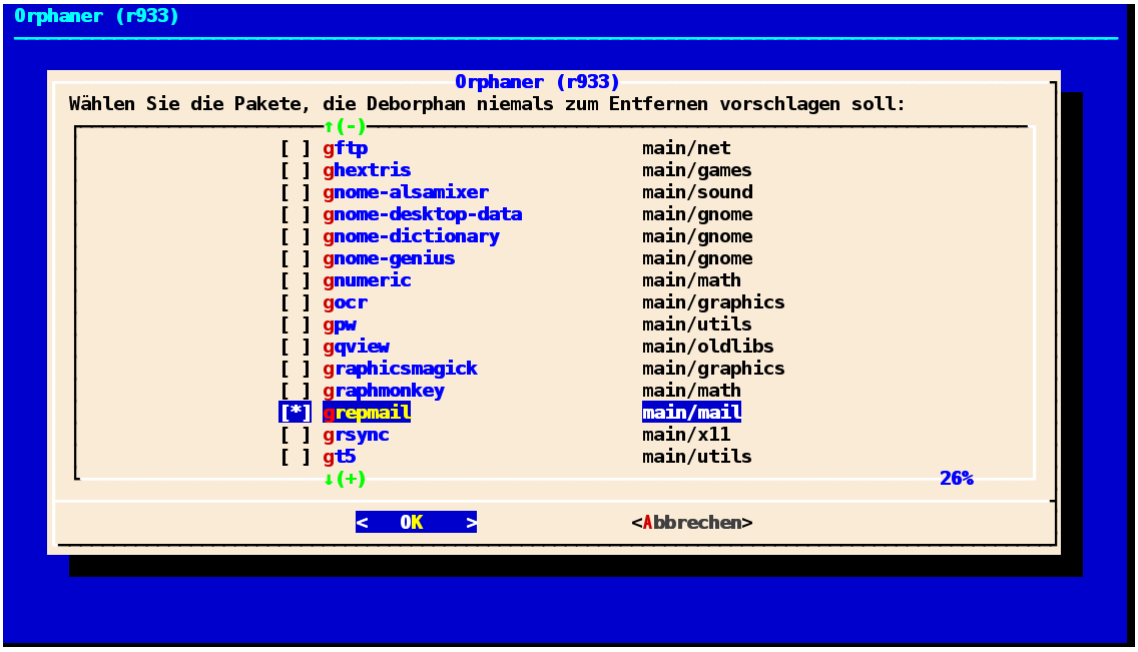

Abbildung 8.24: `editkeep` im Einsatz

8.42.5 `gtkorphan`

`gtkorphan` [Gtkorphan] ist ein graphisches Programm auf der Basis von GTK, welches `deborphan` (siehe Abschnitt 8.42.3) direkt ansteuert. Die Ausgaben stammen daher direkt von `deborphan` und somit aus der Paketbeschreibung.

In der Mitte sehen Sie die Paketliste, wobei Sie über den Reiter zwischen der Darstellung für die verwaisten und nicht verwaisten Pakete umschalten können. Für jeden Eintrag ist der Paketname (siehe Abschnitt 2.11), die Paketgröße, der Distributionsbereich (siehe Abschnitt 2.9), die Paketkategorie (siehe Abschnitt 2.8) sowie die Paketpriorität (siehe Abschnitt 2.13) aufgeführt.

Als zusätzliche Optionen ergänzen Sie die Liste einerseits um bereits gelöschte Pakete, von denen aber noch Konfigurationsdateien vorhanden sind, und andererseits um Pakete aus allen anderen Paketkategorien (siehe Abschnitt 2.8). Um den bereits weiter oben angesprochenen Ratemodus zu verwenden, wählen Sie im Auswahlfeld den gewünschten Eintrag aus der Liste der Möglichkeiten aus. Mit einem Klick auf OK werden alle Waisen von ihrem System entfernt, die Sie zuvor aus der Paketliste ausgewählt haben.

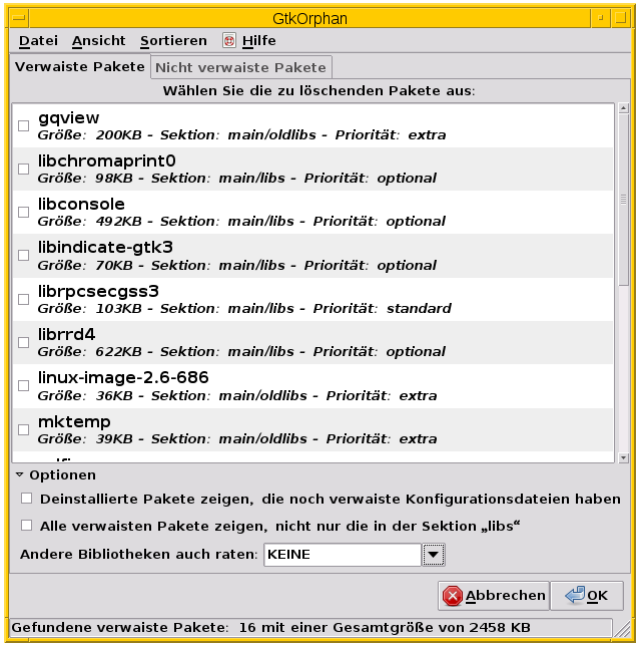

Abbildung 8.25: `gtkorphan` bei der Arbeit

8.42.6 aptsh

Die Terminalemulation `aptsh` (siehe Abschnitt 6.2.3) verfügt über die integrierten Kommandos `orphans` und `orphans-all`, mit denen Sie ebenso Waisen aufspüren können. Grundlage sind die Ergebnisse, die `deborphan` mit Hilfe der Paketbeschreibungen liefert.

Während das erstgenannte Kommando nur nach vereinsamten Bibliotheken sucht und analog zu `deborphan` arbeitet, entspricht `orphans-all` eher dem Aufruf `deborphan -a` und bezieht alle installierten Pakete in die Suche mit ein. Als Ergebnis erhalten Sie eine unsortierte Liste, bei letzterem Kommando zweispaltig mit der Aufteilung aus Distributionsbereich (siehe Abschnitt 2.9) und Kategorie (siehe Abschnitt 2.8) (linke Spalte) sowie dem Paketnamen (siehe Abschnitt 2.11) (rechte Spalte). Die nachfolgenden Abbildungen stammen aus einem xubuntu 13.04 und zeigen die beiden Aufrufe.

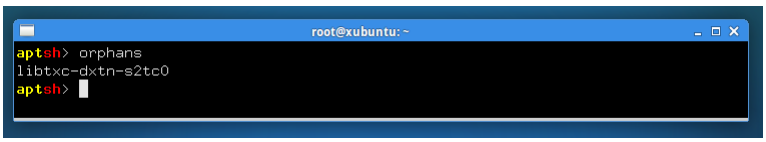

Abbildung 8.26: `aptsh` mit der Ausgabe des Kommandos `orphans`

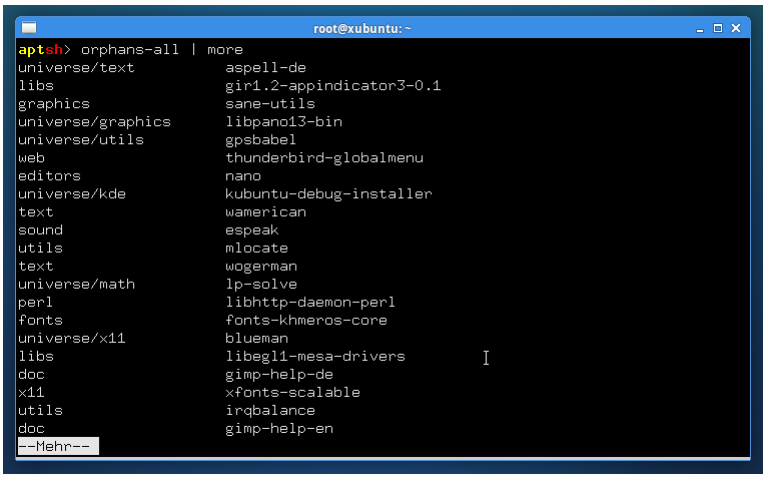

Abbildung 8.27: `aptsh` mit der Ausgabe des Kommandos `orphans-all`

8.42.7 `wajig`

Ähnlich wie die `aptsh` verfügt `wajig` [Debian-Paket-wajig] über Kommandos zur Suche nach Waisen – `orphans` und `list-orphans`. Beide liefern Ihnen das gleiche Ergebnis. Möglich ist ein Aufruf mittels `wajig orphans` oder die Eingabe des Kommandos in der `wajig`-Shell. Damit listet es die Bibliotheken auf, die nicht (mehr) von einem installierten Paket benötigt werden. Andere Pakete werden bei der Recherche nicht berücksichtigt.

Die Analyse basiert auf dem Werkzeug `deborphan` (siehe Abschnitt 8.42.3). Daher muss das entsprechende Paket installiert sein, wenn Sie dieses Kommando verwenden möchten. Abbildung 8.28 zeigt das Ergebnis der Suche nach Waisen in der `wajig`-Shell.

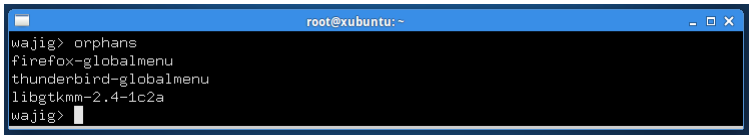

Abbildung 8.28: `wajig` mit der Ausgabe des Kommandos `orphans`

8.43 Paketoperationen erzwingen

Die vorgestellten Werkzeuge zur Paketverwaltung sind als sehr pingeling bekannt, um den Zustand Ihres Systems möglichst stabil und benutzbar zu halten. Dazu zählen beispielsweise eine saubere Installation, keine Konflikte zwischen den installierten Paketen, das Einspielen von Aktualisierungen und Patches sowie keine offenen Paketabhängigkeiten.

Trotz dieser Qualitätskontrolle können Dinge schiefgehen. Dazu zählen beispielsweise unschöne Paketkonflikte — zwei Pakete bedingen einander und lassen sich nicht nacheinander installieren. In dieser Situation hilft Ihnen Spezialwissen weiter, zu dem die Möglichkeiten von `dpkg`, `apt` und `aptitude` gehören, um Sicherheitschecks und Warnungen zu ignorieren und Aktionen trotzdessen durchzuführen. Bitte behalten Sie dabei stets im Hinterkopf, dass diese Schritte und Optionen Ihr System auch unbenutzbar machen können.

ToDo

- Details: `dpkg --force-help`
  - warnen, aber fortsetzen: `--force-`
  - mit Fehler anhalten: `--no-force-` und `--refuse-`

- `dpkg`-Schalter (Auswahl):
  - `bad-verify`: Paket installieren, selbst wenn Authentizitätsüberprüfung misslingt
  - `overwrite` : Datei eines anderen Pakets überschreiben
  - `confmiss` : Fehlende Konf.-Dateien immer installieren
  - `conflicts` : Installation kollidierender Pakete erlauben
  - `depends` : Alle Abhängigkeitsprobleme in Warnungen umwandeln
  - `remove-essential`: Ein essenzielles Paket entfernen
  - `--force-architecture`: Paket installieren, welches nicht zur Architektur passt
  - `--force-remove-reinstreq`:

- `apt`
  - Option `-f`
- in `/var/lib/dpkg/status` herumpfuschen
  - siehe https://tipstricks.itmatrix.eu/force-aptitudeapt-get-ingoring-broken-dependencies/

8.44 Paketstatusdatenbank reparieren

`dpkg` führt permanent Buch über alle Pakete auf einem Debian-System. Es merkt sich in seiner Paketstatusdatenbank den Zustand jedes einzelnen `deb`-Pakets, welches es jemals in den Fingern hatte (siehe dazu auch Abschnitt 8.4 und Abschnitt 8.5). Die Paketstatusdatenbank befindet sich in der Datei `/var/lib/dpkg/status`.

Daraus ersehen Sie ganz eindeutig, ob ein Paket schon einmal installiert war, ob alle notwendigen Schritte vollständig und fehlerfrei abgelaufen sind, ob das Paket auf `hold` gesetzt wurde, ob es wieder entfernt wurde und ob bspw. noch Reste aus dem Paket auf ihrem System verblieben sind. Zu letzterem zählen z.B. die Konfigurationsdateien eines Pakets.

8.44.1 Bit-Dreher reparieren

Die Einträge in der Paketstatusdatenbank sind 7-Bit-Werte, d.h. das achte Bit ist nicht gesetzt. Mittlerweile ist `dpkg` recht robust gegen unbekannte Felder, auch wenn diese 8-Bit-Werte enthalten. Trotzdem funktioniert vieles nicht mehr, wenn die Paketstatusdatenbank außerhalb des Formates gemäß der Spezifikation nach RFC 822 vorliegt [RFC822].

Die Ursache für dieses gedrehte Bit kann sowohl ein Hardware-Crash sein, während `dpkg` werkelte, als auch ein Bit-Dreher auf dem Speichermedium selbst. Das nachfolgende Beispiel demonstriert das anhand des Eintrags zum Paket *geekcode*. In Zeile 10 liegt ein solcher Bitfehler vor – das achte Bit für den Doppelpunkt `:` ist hier gesetzt.

Bit-Dreher in der Datei /var/lib/dpkg/status für das Paket geekcode (Ausschnitt)

```
Package: geekcode
Status: install ok installed
Priority: optional
Section: games
Installed-Size: 166
Maintainer: Eric Dorland <eric@debian.org>
Architecture: amd64
Multi-Arch: foreign
Version: 1.7.3-6
Depends\textdegree{} libc6 (>= 2.7)
Description: Program for generating geekcode
 This is a program for generating the geekcode.
 See http://www.geekcode.com for more info and for discovering
 if you need the geekcode.
Homepage: http://sourceforge.net/projects/geekcode/
```

Verarbeitet `dpkg` diese Daten, kann es mit diesem Wert nichts anfangen und bringt seine Verärgerung darüber mit einer deutlichen Fehlermeldung zum Ausdruck. Dieser Fall ist noch vergleichsweise leicht zu reparieren, indem Sie das ISO-Latin-1-Zeichen ° mit Hilfe ihres Texteditors wieder durch einen Doppelpunkt ersetzen. Daraufhin ist `dpkg` wieder glücklich.

dpkg bricht nach einem Bit-Dreher in der Statusdatenbank ab

```
# dpkg --configure --pending
dpkg: error: parsing file '/var/lib/dpkg/status' near line 9 package 'geekcode':
 field name `Depends°' must be followed by colon
```

8.44.2 Die Paketstatusdatenbank aus dem lokalen Backup wiederherstellen

Anspruchsvoller wird es jedoch bspw. dann, wenn mehr als nur ein einzelnes Bit oder Byte kaputt ging, plötzlich ganze Blöcke fehlen, oder sich diese nach einer Reparatur des Dateisystems in ganz anderen Formaten in dieser Datei wiederfinden. Dann hilft meist nur noch ein Wiederherstellen der Paketstatusdatenbank aus ihrem Backup.

Zum Glück gibt es auf einem Debian-System mehrere Backups der Paketstatusdatenbank. Ging die Datei erst gerade eben kaputt, so ist die Chance sehr hoch, dass die vorhergehende Sicherheitskopie namens `/var/lib/dpkg/status-old` noch intakt ist. Dieses Backup entspricht dem Zustand der Paketstatusdatenbank *vor der letzten Änderung* und hinkt somit nur um eine einzige Aktion hinterher. Kopieren Sie diese zurück nach `/var/lib/dpkg/status`, so fehlt `dpkg` nur das Wissen über die letzte Installation oder Deinstallation eines Pakets. Führen Sie diese letzte Aktion erneut durch, ist alles gerettet.

Ist hingegen auch `/var/lib/dpkg/status-old` defekt, so gibt es unter `/var/backups/dpkg.status.*` tlw. komprimierte Schnappschüsse der Paketstatusdatenbank. Diese beziehen sich auf die letzten sieben Tage, an denen ihr Rechnersystem eingeschaltet war.

Ersetzen Sie die aktuelle Paketstatusdatenbank durch eine ältere Version, so weiß `dpkg` nichts mehr von sämtlichen Aktionen, die Sie seitdem durchgeführt haben. Insbesondere denkt es, ein Paket sei noch installiert, wenn es seit dem Backup der Datei deinstalliert wurde. In diesem Fall führen Sie die Entfernung des entsprechenden Pakets nochmals durch. Schlägt diese Aktion fehl, weil die dazu erwarteten Dateien bereits nicht mehr da sind, so kann es helfen, das entsprechende Paket mittels `dpkg -i` vorher nochmals zu installieren, um wieder einen konsistenten Zustand zu erreichen.

8.44.3 Die Paketstatusdatenbanken von APT und `aptitude`

Auch APT und `aptitude` führen eigene Archive über ihre Aktionen, zu denen ebenfalls unter `/var/backups/` tägliche Schnappschüsse existieren. Allerdings ist ein Verlust dieser Statusdatenbanken weniger kritisch, da die meisten Informationen darin durch ein `apt-get update` bzw. `aptitude update` schnell wiederherstellbar sind. Grundlage dafür sind jedoch die Daten aus der Paketstatusdatenbank von `dpkg`.

Das einzige, was auf diese Art und Weise nicht wiederhergestellt werden kann, sind APT- und `aptitude`-spezifische Informationen. Dazu zählen bspw. die Markierungen *automatisch installiert* sowie die Vormerkungen und User-Tags von `aptitude` (siehe Kapitel 11).

8.45 Distribution aktualisieren (update und upgrade)

8.45.1 Vorworte

Das Aktualisieren einer bestehenden Linuxinstallation ist immer eine etwas heikle Geschichte und eine Frage des Selbstvertrauens sowie des Bauchgefühls. Es geht dabei schließlich nicht nur an vergleichbaren Kleinkram wie ein einzelnes Paket, sondern um das ganze System, in dessen Pflege Sie bereits viel Zeit und Mühe gesteckt haben. Dieser Aufwand soll schließlich nicht umsonst gewesen sein.

Eine Aktualisierung sind stets größere Umbauarbeiten, bei dem sich vergleichsweise viel ändert und durchaus auch etliches schief gehen kann, womit Sie nicht unbedingt rechnen. An der Stelle sei jedoch zu Ihrer Beruhigung angemerkt, dass der Wechsel von Debian 6 *Squeeze* auf Debian 7 *Wheezy* recht unspektakulär verlief und vielfach problemlos über die Bühne ging. Ähnliches gilt für den Wechsel auf den Debian 8 *Jessie*.

Trotzdem halten wir es für ganz praktisch, wenn wir Ihnen eine Schritt-für-Schritt-Abfolge zur Verfügung stellen, der Sie folgen können. Das verringert die Wahrscheinlichkeit, dass bei der Aktualisierung etwas vergessen wird. Empfehlenswert ist auch, den Vorgang zu zweit mit einem Sparringspartner vorzunehmen. Das mindert die Anspannung und hilft Situationen zu umschiffen, in denen etwas Unbekanntes auftritt, wo Sie vielleicht allein nicht ohne Hilfe weiterkommen.

Desweiteren sind mehrere Hilfsmittel von Nutzen. Dazu gehören neben einem vollständigen und verfügbaren Backup Ihrer Daten eine CD/DVD oder ein USB-Stick mit einem Live-System für alle Fälle, um Ihr System bei Missgeschicken davon booten zu können und darüber wieder Zugriff auf Ihr System zu erhalten. Ein weiteres Gerät mit Internetzugang hilft dabei, Lösungen zu aufkommenden Fragen oder Unklarheiten zu recherchieren. Stift und Papier klingen trivial, ermöglichen aber flinke Notizen, falls das doch erforderlich sein sollte.

8.45.2 Vom `upgrade` zum `dist-upgrade`

Die vollständige Aktualisierung des Paketbestands erfolgt mit dem APT-Unterkommando `dist-upgrade`. Es ist auf den ersten Blick sehr ähnlich zu `upgrade`, es bestehen jedoch wesentliche Unterschiede zwischen beiden. Ersteres bezieht nur Änderungen innerhalb der bestehenden Veröffentlichung, das Zweite bezieht alles von der neuen Veröffentlichung.

8.45.3 Unsere empfohlene Reihenfolge

Wir empfehlen Ihnen, bei der Aktualisierung Ihrer Distribution die nachfolgenden Schritte nicht außer Acht zu lassen.

1. Lesen Sie zuerst die Dokumentation und die Hinweise zum Distributionswechsel. Darin ist beispielsweise beschrieben, welche Veränderungen Sie bezüglich interner Strukturen und Dienste erwartet. Diese Informationen finden Sie unter dem Stichwort Veröffentlichungshinweise – auf englisch *Release Notes* – einerseits auf der Webseite des Debian-Projekts [Debian-Release-Notes] sowie als Bestandteil der offiziellen, verfügbaren Debian-Images.

2. Halten Sie Ihre Zugangsdaten für administrative Zwecke bereit.

3. Sofern noch nicht vorhanden, erzeugen Sie ein Backup Ihrer wichtigen Daten auf ein möglichst externes Medium. Dazu zählen neben den Nutzerdaten insbesondere die Konfigurationseinstellungen Ihrer Programme. Häufig werden dabei Inhalte von Datenbanksystemen und Webpräsenzen übersehen, die sich unter dem Verzeichnis `/var` tummeln. Überprüfen Sie danach Ihr Backup auf Vollständigkeit. Nicht ist enttäuschender als eine Datensicherung, welche sich im Nachhinein als unvollständig herausstellt.

4. Setzen Sie die Veröffentlichung *testing* oder *unstable* ein, fahnden Sie mit Hilfe des Pakets `apt-listbugs` [Debian-Paket-apt-listbugs] nach möglicherweise kritischen Fehlern in der Debian-Fehlerdatenbank (siehe Abschnitt 32.3.2).

5. Aktualisieren Sie die bestehende Paketliste mittels `apt-get update` (siehe Abschnitt 3.13). Damit bringen Sie die Paketliste auf den aktuellen Stand und verringern die Unterschiede zum verfügbaren Paketbestand.

6. Spüren Sie Waisen und nicht mehr benötigte Pakete mittels `apt-get autoremove` auf (siehe Abschnitt 8.42). Dieser Schritt verringert den zu berücksichtigenden Paketbestand und macht sich in mehrfacher Hinsicht bemerkbar. Einerseits werden Altlasten beseitigt, sie sparen Zeit und Festplattenplatz, es müssen somit weniger Datenpakete über die Leitung geschubst werden und anderseits danach eine geringere Anzahl Pakete aktualisiert werden.

7. Spielen Sie die letzten Paketversionen Ihrer aktuell genutzten Veröffentlichung mittels `apt-get upgrade` ein. Damit verringern Sie die Unterschiede zum Versionswechsel weiter.

8. Passen Sie Datei `/etc/apt/sources.list` entsprechend auf die neue Distribution an. Wechseln Sie bspw. von Debian 7 *Wheezy* auf Debian 8 *Jessie*, ändern Sie alle Vorkommen von `wheezy` auf `jessie`.

9. Bringen Sie die Paketliste mittels `apt-get update` auf den neuesten Stand (siehe Paketliste aktualisieren unter Abschnitt 3.13).

10. Aktualisieren Sie die Distribution mittels `apt-get dist-upgrade`. Jetzt wird der Distributionswechsel vollzogen und alle bestehenden Pakete werden erneuert, sofern neue Varianten vorliegen.

8.45.4 Anmerkungen

Ein Distributionswechsel ist auch mit `aptitude` möglich. Dazu verwenden Sie in Schritt 10 obiger Liste auf der Kommandozeile statt `apt-get dist-upgrade` den Aufruf `aptitude full-upgrade`. Aus historischen Gründen besteht noch ein Synonym zu `dist-upgrade`, welches Sie derzeit ebenfalls noch benutzen können.

Über die Textoberfläche gelingt Ihnen gleiches nur über einen kleinen Umweg. Dazu markieren Sie zunächst mittels Aktionen → Aktualisierbare markieren alle Pakete, für die eine neuere Variante verfügbar ist (Kurzform: Taste **U**). In Folge lösen Sie mittels **g** die Erneuerung der zuvor markierten Pakete aus.

Kapitel 9

Dokumentation

Ein einzelnes Werk, welches die Debian-Paketverwaltung in allen seinen Facetten behandelt, gibt es unseres Wissens bisher nicht. Das Know-How dazu ist über diverse Quellen in unterschiedlichen Qualitätsstufen verstreut.

Um flink ein Kommando oder eine Option nachzuschlagen, geht daher der erste Griff zu den `man`- und `info`-pages der Werkzeuge `dpkg`, APT und `aptitude`. Diese Hilfeseiten sind zwar meist nur eine Kurzfassung der komplexen Werkzeuge, helfen aber im Bedarfsfall trotzdem weiter. Der nicht zu unterschätzende Vorteil besteht darin, dass Ihnen diese Informationen stets auf jedem Debian-System zur Verfügung stehen. Diese Hilfedokumente sind Bestandteil der Pakete und werden mitgeliefert. Ausgegliederte Pakete mit Dokumentation installieren Sie bei Bedarf einfach nach.

Nummer zwei ist der Griff zur Dokumentation, welche im Verzeichnis `/usr/share/doc/` auf ihrem System liegt. Für APT und `aptitude` ist diese Dokumentation jeweils als separates Paket ausgelagert. Diese Pakete heißen *apt-doc* sowie *aptitude-doc*. Genauer gehen wir dazu in den beiden Abschnitten zu `apt-doc` in Abschnitt 9.2 und zum `aptitude`-Handbuch in Abschnitt 9.5 ein.

Darüberhinaus werfen wir einen Blick auf weitere, passende Online- und Offline-Dokumentation zu diesem Thema. Diese Liste erhebt keinen Anspruch auf Vollständigkeit und soll Ihnen nur als Anregung dienen und zeigen, was uns als lesenswert erscheint und sich zur Ergänzung mit dem vorliegenden Werk eignet. Über den Tellerrand hinausblicken schadet nie.

9.1 Die apt-dpkg-Referenzliste

Matthew Danish hat zu den beiden Programmen `dpkg` und APT eine Referenzliste zusammengestellt. Diese ist als offizielles Debianpaket mit dem Namen *apt-dpkg-ref* [Debian-Paket-apt-dpkg-ref] verfügbar. Derzeit existiert diese Übersicht nur in englischer Sprache, Übersetzungen in andere Sprachen liegen bislang nicht vor.

Installieren Sie dieses Paket über die Paketverwaltung, finden Sie diese Dokumentation danach im Verzeichnis `/usr/share/doc/apt-dpkg-ref/` wieder. Neben den Lisp- und LaTeX-Quellen ist die Dokumentation in Form einer HTML-, PDF- und PostScript-Datei verfügbar.

Alle Formate beinhalten eine Übersicht zu den grundlegenden `dpkg`- und APT-Kommandos samt griffiger Beschreibung der einzelnen Optionen. Zudem werden kurz und knapp die Beziehungen zwischen `dpkg` und APT hergestellt. Eine kurze Einführung zum Bauen von `deb`-Paketen aus den Paketquellen rundet die Beschreibung ab. Abbildung 9.1 zeigt einen Ausschnitt der Referenzliste im PDF-Betrachter *xpdf*.

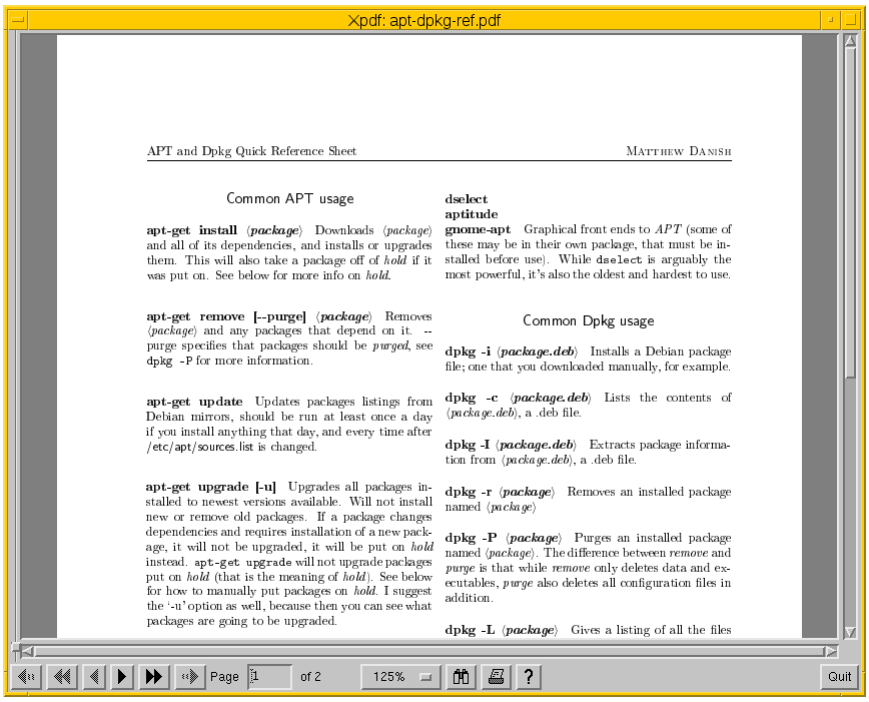

Abbildung 9.1: Die apt-dpkg-Referenzliste im PDF-Betrachter *xpdf*

9.2 `apt-doc` — das Benutzerhandbuch zu APT

Das Paket *apt-doc* [Debian-Paket-apt-doc] beinhaltet das Benutzerhandbuch zu APT. Es wurde bereits 1998 von Jason Gunthorpe begonnen und zwischenzeitlich mehrfach aktualisiert und in diverse Sprachen übersetzt. Es steht Ihnen als HTML- und Textversion in englisch, deutsch, spanisch, französisch, polnisch und portugiesisch zur Verfügung. Alle Übersetzungen finden Sie im gleichen Paket.

Nach der Installation finden Sie die deutsche Dokumentation als HTML-Dokument im Verzeichnis `/usr/share/doc/apt-doc/guide.de.html/` wieder. Zur Nutzung von APT in einem „Turnschuhnetzwerk“ (offline) hilft Ihnen die Beschreibung unter `/usr/share/doc/apt-doc/offline.de.html/index.html` weiter.

Es umfasst eine Einführung in die Paketverwaltung und beschreibt recht knapp die Werkzeuge `dselect`, `apt-deselect` und APT sowie deren verschiedene Aufrufparameter. Nützlich sind in der Dokumentation die Ausgaben der Programme im Terminal, die Ihnen auch dabei helfen, die diversen Statusanzeigen und Fehlermeldungen der Programme zu überblicken und zu verstehen. Abbildung 9.2 zeigt Ihnen dazu einen Ausschnitt der Dokumentation im Webbrowser Iceweasel an.

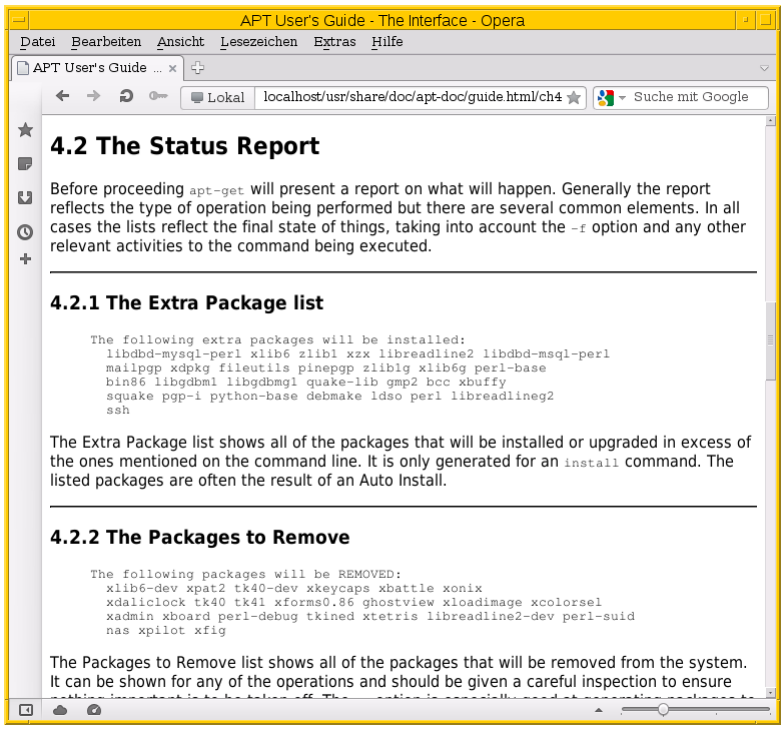

Abbildung 9.2: Ausschnitt aus der APT-Dokumentation mit `apt-doc`

Pflege ohne Internetzugang
Hier im Buch beschäftigen wir uns ausführlicher mit diesem Thema in Kapitel 37.

9.3 APT-Spickzettel von Nixcraft

Im Nixcraft-Blog [nixcraft-blog] finden Sie eine Übersicht, welches die beiden Programme `dpkg` und `apt-get` mit vielen Erklärungen und aussagekräftigen Beispielen gegenüberstellt. Zu beiden Programmen gibt es einen Spickzettel für den Webbrowser (auf englisch *cheat sheet*), welcher Ihnen auch in vielen Situationen weiterhilft. Der Spickzettel steht zu `dpkg` [nixcraft-dpkg] und `apt-get` [nixcraft-apt-get] zur Verfügung. Letzteres zeigt Ihnen Abbildung 9.3.

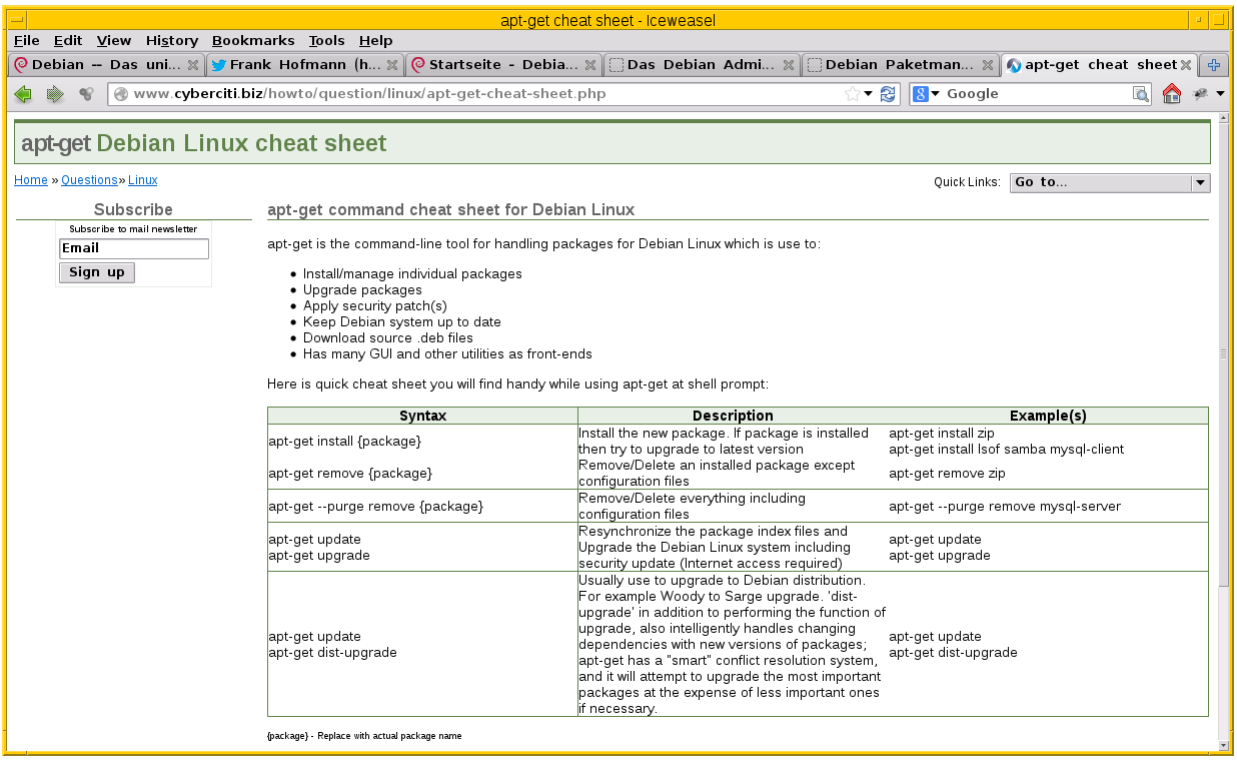

Abbildung 9.3: Spickzettel zu `apt-get`

9.4 Pacman Rosetta

Eine ausführliche Zusammenstellung der Kommandozeilenparameter und Funktionen populärer Paketverwaltungen beinhaltet die Pacman Rosetta [Pacman-Rosetta] aus dem Wiki zu Arch Linux. Anhand von Aufrufen aus der Praxis stellt es Pacman (Arch Linux), YUM (RedHat/Fedora), `dpkg` und `apt-get` (Debian, Ubuntu) sowie `rug` (älteres Suse), Zypper (openSUSE) und `emerge` (Gentoo Linux) gegenüber. Eine ähnliche Übersicht mit den aktuellen Parametern finden Sie auch im Anhang unter „Kommandos zur Paketverwaltung im Vergleich“ Kapitel 46.

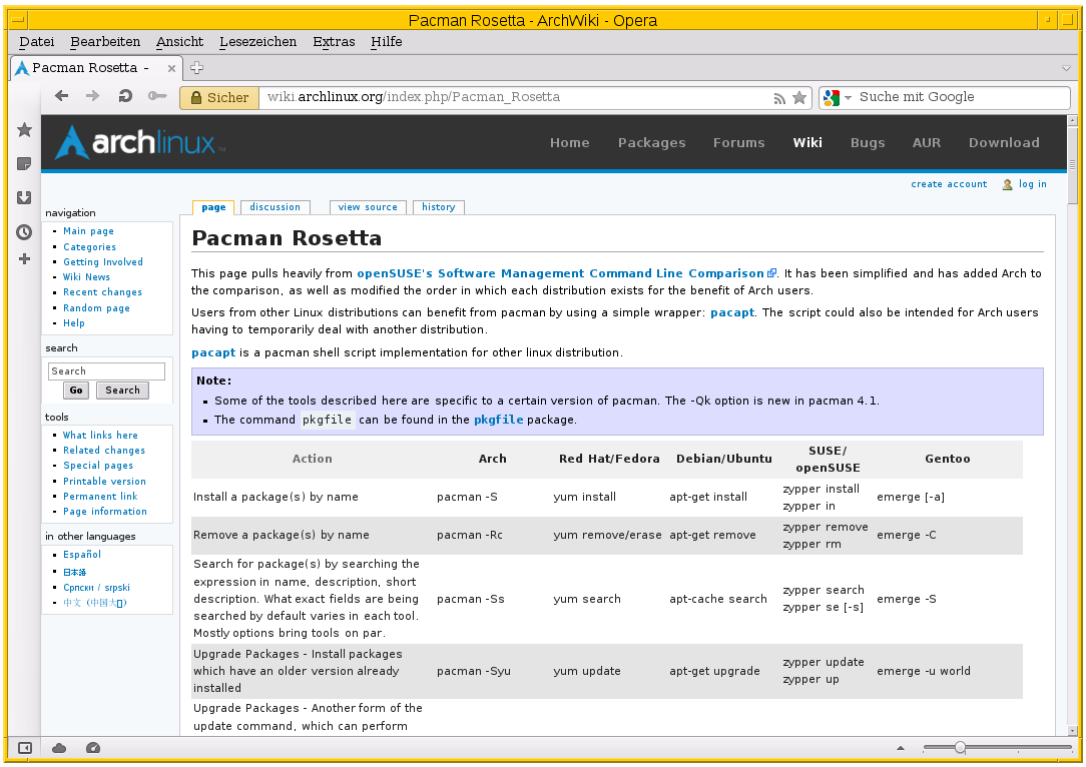

Abbildung 9.4: Die Pacman Rosetta (Ausschnitt)

9.5 Handbuch zu `aptitude`

`aptitude` ist ein recht komplexes Programm und bedarf daher einiges an Dokumentation. Diese steht bislang in 9 Sprachen als HTML-Datei bereit, so in englisch, tschechisch, spanisch, finnisch, französisch, italienisch, niederländisch, russisch und japanisch. Eine deutsche Übersetzung fehlt leider bisher noch.

Seit 2013 ist die Zusammenstellung für `aptitude` in der Version 0.6.8.2 in 7 Sprachen verfügbar [aptitude-dokumentation], für Debian 8 *Jessie* kamen zudem niederländisch und russisch dazu. Zuvor war es längere Zeit recht ruhig um die Dokumentation, da der bisherige Maintainer nicht mehr erreichbar war. Nachdem das neue `aptitude`-Team vollen Zugriff auf die Daten bei Alioth – Debians FusionForge-Installation – hatte, ging es flink voran. Leider verweisen etliche Suchmaschinen auch heute noch auf die vorherige Version 0.4.11.2 aus dem Jahr 2008 [aptitude-dokumentation-veraltet].

Die Dokumentation ist auch als sprachspezifisches Debianpaket verfügbar. Bspw. enthält *aptitude-doc-en* die englische und *aptitude-doc-fr* die französische Übersetzung.

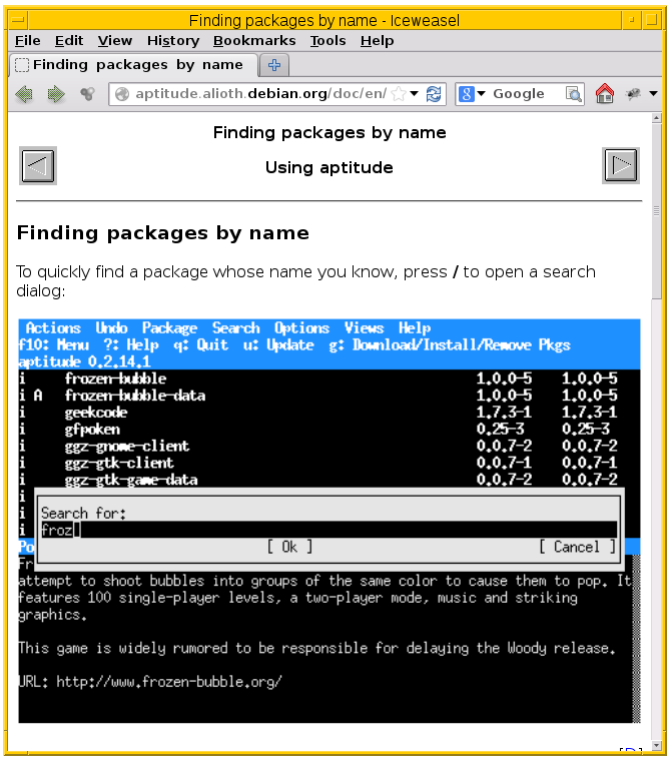

Abbildung 9.5: Dokumentation zu `aptitude`

9.6 The Debian Administrator's Handbook

Seit 2003 setzen Raphaël Hertzog und Roland Mas mit ihrem Kompendium „The Debian Administrator's Handbook“ Maßstäbe [Hertzog-Mas-Debian-Administrators-Handbook] (französischer Originaltitel: „Cahiers de l'Admin“). Das Buch spielt in der gleichen Liga wie Frank Ronneburgs „Debian-Anwenderhandbuch“ [Debian-Anwenderhandbuch], Michael Koflers „Linux – das umfassende Handbuch“[1] [Kofler-Linux-2013] und Martin Kraffts Buch „Das Debian-System. Konzepte und Methoden“ [Krafft-Debian-System]. Aufgrund seiner Einzigartigkeit im französischen Sprachraum erreichen Neuauflagen regelmäßig einen Spitzenplatz in den Verkaufslisten des Buchhandels.

Ursprünglich nur in französischer Sprache verfasst, steht es mittlerweile auch in einer englischen und spanischen Übersetzung bereit. Diese drei Varianten sind sowohl als kostenfreie, digitale Version (PDF und eBook), als auch als kostenpflichtige, gedruckte Variante (Paperback, Print on demand) erhältlich. Die Finanzierung der Übersetzung ins Englische erfolgte über eine Crowdfunding-Kampagne. Übersetzungen in andere Sprachen wie bspw. Arabisch, Farsi, Deutsch, Griechisch und Russisch werden von Freiwilligen beigesteuert, sind aber derzeit noch nicht ganz vollständig und daher nicht in gedruckter Form verfügbar.

Das Buch wird jeweils für die aktuelle Debian-Veröffentlichung angepaßt. Gleichzeitig entsteht daraus auch ein reguläres Debianpakete namens *debian-handbook* [Debian-Paket-debian-handbook] welches in die Distribution wieder einfließt. Das Paket beinhaltet derzeit den französischen, englischen und spanischen Text.

Das Buch ist für alle Anwender gedacht, die Debian GNU/Linux-Systeme administrieren. Daher deckt es neben einem Einblick in das Debian-Projekt (siehe Abschnitt 1.1) alle Bereiche ab, über Sie als Administrator Bescheid wissen müssen – die Debian-Installation, die Einrichtung und Betreuung von Diensten wie KVM, Xen und LXC sowie die Absicherung der von Ihnen betreuten Systeme. Auch die Thematik Automatisierung kommt nicht zu kurz, bspw. mittels FAI (siehe Kapitel 30). In Bezug auf das Debian-Paketformat (siehe Kapitel 4) setzen die Autoren auf `dpkg` (Grundlagen und Ebenen) und zeigen das Paketmanagement anhand von `aptitude` und `synaptic`. Auch der Erstellung von Debianpaketen ist ein eigenes Kapitel gewidmet.

[1] das Werk basiert von jeher auf SuSE-Linux, reißt aber alle Bereiche des Linux-Alltags mit an

Abbildung 9.6: Cover des Buches

9.7 Weitere Bücher

Neben bereits oben besprochenen Dokumenten existieren viele thematisch breiter angelegte Werke und Veröffentlichungen. Diese verschaffen dem Neueinsteiger eine sehr gute Einordnung des Themas Paketmanagement im Kontext der Einführung in den Umgang mit Linux als Betriebssystem sowie der allgemeinen Betreuung von Rechnern und ganzen Rechnerverbünden. Auch als dauerhaftes Nachschlagewerk sind diese Bücher wirklich zu empfehlen, denn diese Bücher haben Substanz – auch wenn diese zum Teil bereits etwas älter sind. Die meisten Autoren konzentrieren sich dabei auf das Basiswissen mit `dpkg` in Kombination mit APT, `aptitude` oder `synaptic`:

- Frank Ronneburg „Das Debian-Anwenderhandbuch“ – 1999 bis 2008 als Buch aufgelegt und danach nur noch als Online-Dokumentation weitergeführt [Debian-Anwenderhandbuch]. Fokussiert ausschließlich auf Debian GNU/Linux.
- Martin F. Krafft „Das Debian-System. Konzepte und Methoden“ – 2006 als Buch auf der Basis von Debian 3.1 *Sarge* veröffentlicht. Eine Fortsetzung und Aktualisierung ist bislang nicht erfolgt [Krafft-Debian-System]. Fokussiert ausschließlich auf Debian GNU/Linux.

- Michael Kofler „Linux – das umfassende Handbuch“ – Erstveröffentlichung 1995 mit jährlicher Aktualisierung. Nicht distributionsspezifisch, sondern für `deb`- und `rpm`-basierte Linux-Distributionen gleichermaßen gut geeignet [Kofler-Linux-2013].
- Heike Jurzik „Debian GNU/Linux: Das umfassende Handbuch“ – Erstveröffentlichung als Buch 2006 zu Debian 3.1 *Sarge*, danach regelmäßige Aktualisierung und Erweiterung [Jurzik-Debian-Handbuch]. In der Regel erfolgt eine weitere Veröffentlichung, wenn eine neue Debian-Veröffentlichung freigegeben wurde. Das Buch richtet sich an Ein- und Umsteiger, die ihr Wissen Stück für Stück erweitern – vom Einstieg über die Installation bishin zum Betrieb eigener Dienste und Server. Fokussierung auf `deb`-basierte Distributionen mit dem Schwerpunkt Debian GNU/Linux.
- Eric Amberg „Linux-Server mit Debian 6 GNU/Linux: Das umfassende Praxishandbuch“ – erschienen 2012 zu Debian 5 *Lenny* und Debian 6 *Squeeze*, im Februar 2014 fortgesetzt mit einer aktualisierten Fassung für Debian 7 *Wheezy* [Amberg-Linux-Server-Praxishandbuch]. Fokussiert ausschließlich auf Debian GNU/Linux.

Teil III

Praxis

Kapitel 10

APT und `aptitude` auf die eigenen Bedürfnisse anpassen

10.1 Konfigurationsdateien von APT

APT und auch alle Programme, die APTs Bibliotheken benutzen, greifen auf diverse Konfigurationsdateien zu. Über diese Dateien werden unterschiedliche Funktionen abgebildet.

Alle Varianten der Konfigurationsdateien gibt es in zwei Geschmäckern:

- als einzelne Datei — diese ist typischerweise dem lokalen Systemadministrator vorbehalten
- als Verzeichnis mit dem Suffix `.d`. Alle Dateien in einem solchen Verzeichnis, die bestimmten Regeln hinsichtlich dem Dateinamen entsprechen, werden so interpretiert, als wären sie an der gleichnamigen Datei ohne `.d` angehängt (siehe u.a. Man-Page `run-parts(8)`. Insbesondere dürfen keine Punkte im Dateinamen vorkommen. Diese Verzeichnisse werden typischerweise von Paketen verwendet, um paketspezifische Konfigurationsschnipsel darin abzulegen. Auch Sie als lokaler Systemadministrator legen darin Dateien ab, bspw. über das Konfigurationsmanagement.

10.1.1 Verhalten von APT und Hooks: /etc/apt/apt.conf(.d)

In der Datei `/etc/apt/apt.conf` bzw. in den Dateien im Verzeichnis `/etc/apt/apt.conf.d/` können viele Aspekte von APT und anderen APT-Frontends konfiguriert werden. Alle diese Einstellungen überschreiben Sie bei Bedarf explizit auch auf der Kommandozeile. Dazu verwenden Sie als Schalter `-o`, gefolgt von der Form `<Schlüssel>=<Wert>`. Darunter fallen u.a.:

- wie oft APT selbstständig Paketlisten und Pakete herunterladen soll (alle Einstellungen, die mit `APT::Periodic::` beginnen),
- ob empfohlene (*Recommends*) oder vorgeschlagene Pakete (*Suggests*) per Default installiert werden sollen,
- welche Pakete nie automatisch entfernt werden sollen,
- ob Paketlisten lokal entpackt oder komprimiert gespeichert werden sollen,
- welche Kompressionsalgorithmen beim Herunterladen von Paketlisten bevorzugt werden sollen,
- wohin APT Paketlisten und Pakete herunterladen soll. (Sollte man nicht systemweit ändern, kann aber beim Bearbeiten einer Chroot-Umgebung von außen durchaus als Einstellung auf der Kommandozeile nützlich sein.)
- An- und Abschalten von Fortschrittsbalken (verfügbar ab APT 1.0),
- wo Changelogs heruntergeladen werden können,

- welche Übersetzungen der Paketlisten heruntergeladen werden sollen,
- welche Parameter APT an `dpkg-deb` übergibt,
- Hooks, z.B. vor und nach dem Auspacken von Paketen,
- Konfigurationen für beliebige Tools im APT-Ökosystem, die ebenfalls diese Konfigurationsdateien verwenden, z.B. `aptitude`, `adequate` [Debian-Paket-adequate], `whatmaps` [Debian-Paket-whatmaps], etc.

Um das Vorgehen besser zu verstehen, werfen Sie am besten einen Blick in das Verzeichnis `/etc/apt/apt.conf.d/`. Das nachfolgende Listing zeigt den typischen Inhalt auf einem Debian 8 *Jessie*.

Beispielhafter Inhalt von /etc/apt/apt.conf.d/

```
$ ls /etc/apt/apt.conf.d/
00trustcdrom    10periodic           20packagekit           99synaptic
01autoremove    20apt-show-versions  50unattended-upgrades
10apt-listbugs  20dbus               70debconf
$
```

10.1.2 Konfigurationsdateien von Aptitude

`aptitude` verwendet alle Konfigurationsdateien von APT plus seine eigenen. Diese finden Sie im Verzeichnis `~/.aptitude/` in der Datei `config`. Hier werden auch die interaktiven Änderungen der Konfiguration in der Textoberfläche von `aptitude` gespeichert.

Dabei verwendet die Konfiguration von `aptitude` die gleiche Syntax wie die von APT. Sie kann auch manuell in den Konfigurationsdateien von APT abgespeichert werden, z.B. als Datei `/etc/apt/apt.conf.d/lokale-aptitude-einstellungen`. Diese stören APT nicht, da sie alle mit dem Präfix `Aptitude::` beginnen.

Beispiel einer lokalen Konfigurationsdatei von aptitude

```
aptitude "";
aptitude::Keep-Unused-Pattern "";
aptitude::Delete-Unused-Pattern "";
```

10.1.3 APT-Hooks

- Begriff und Nutzen
  - Ergänzungen, kleine Erweiterungen, Eingriffe
  - standardisierte Abläufe um eigene, paketbezogene Schritte ergänzen
- Festlegung in der APT-Konfiguration
  - wo speichert man das
  - was ist erlaubt, was nicht
  - was sind Gepflogenheiten

10.1.4 cron.daily/apt

- Begriff und Nutzen

10.2 Konfiguration von APT anzeigen

Zu diesem Zweck steht Ihnen das Werkzeug `apt-config` zur Verfügung. Das Ziel des von den Entwicklern gewählten Schnittstellendesigns besteht in der leichten Benutzbarkeit des Programms — auch von Shellskripten aus (siehe [Debian-Wiki-AptConf]).

Die Konfiguration von APT erhalten Sie mit dem Schalter `dump`. Nachfolgend sehen Sie einen Auszug der Ausgabe. Um diese Informationen zusammenzustellen, kombiniert `apt-config` die Inhalte der einzelnen Module zur Konfiguration (siehe dazu Abschnitt 10.1).

Ausgabe der aktuellen Einstellungen von APT mittels apt-config

```
$ apt-config dump
APT "";
APT::Architecture "i386";
APT::Build-Essential "";
APT::Build-Essential:: "build-essential";
APT::Install-Recommends "true";
APT::Install-Suggests "0";
APT::Authentication "";
APT::Authentication::TrustCDROM "true";
APT::NeverAutoRemove "";
APT::NeverAutoRemove:: "^firmware-linux.*";
APT::NeverAutoRemove:: "^linux-firmware$";
APT::NeverAutoRemove:: "^linux-image.*";
APT::NeverAutoRemove:: "^kfreebsd-image.*";
...
$
```

10.3 Interaktives Ändern von Optionen

- `.aptitude/config` (root vs non-root; interaktives Ändern von Optionen) (1)
- Überschreiben von Optionen während des Aufrufs
  - Schalter und Parameter überschreiben Standardwerte
  - wie gebe ich diese beim Aufruf an?

10.4 `aptitude` Format Strings

Format Strings sind eine spezifische Schreibweise einer Ausgabe anhand von Platzhaltern. Sie ähneln der Art und Weise, wie sie in der `printf()`-Funktion in der Programmiersprache C respektive der `print()`-Funktion in Python üblich sind. Eine ausführliche Beschreibung der Platzhalter finden Sie im `aptitude`-Handbuch unter *Customizing the package list* [aptitude-dokumentation-package-list].

Tabelle 10.1 gibt Ihnen eine Übersicht zu den verfügbaren Format Strings. Diese Platzhalter helfen Ihnen in Kombination mit der Suche nach Paketen und bei der Gestaltung der Ausgabe. Sie bestimmen damit, welche Informationen `aptitude` spaltenweise zu einem Paket darstellt.

Tabelle 10.1: Format Strings in `aptitude`

Platzhalter	Bedeutung
`%a`	Flag für die Aktion des Pakets (*Action Flag*)
`%c`	aktueller Paketstatus (*Current State Flag*)
`%d`	die kurze Paketbeschreibung (*description*)

Tabelle 10.1: (continued)

Platzhalter	Bedeutung
`%E`	Name des Source-Pakets
`%I`	die (geschätzte) Installationsgröße (*installed size*)
`%m`	der Name des Paketmaintainers (*maintainer*)
`%M`	gesetzt, falls das Paket automatisch installiert wurde (*Automatic Flag*)
`%p`	Paketname (*package name*)
`%P`	Paketpriorität (*priority*)
`%s`	Bereich, in den das Paket eingeordnet ist (*section*)

Die Voreinstellung von `aptitude` beinhaltet die Platzhalter `%c`, `%a`, `%M`, `%p%` und `%d`. Es umfasst somit die einzelnen Spalten mit dem Paketstatus, der Aktion, das *Automatic Flag*, den Paketnamen und die Paketbeschreibung. Mit zusätzlichen Trennzeichen und dezimalen Angaben vor einem Platzhalter legen Sie die Gestaltung der Ausgabe und die jeweilige Spaltenbreite fest.

Mit der Angabe `%c%a | %15p` erzeugen Sie drei Spalten — den Paketstatus, die Aktion des Pakets und den Paketname — wobei zwischen der zweiten und dritten Spalte ein senkrechter Trennstrich eingefügt wird, der davor und danach noch ein Leerzeichen zur besseren Lesbarkeit umfaßt. Die dritte Spalte hat eine feste Breite von 15 Zeichen, während die anderen Spalten variabel bleiben.

Im Aufruf von `aptitude search` geben Sie die Platzhalterfolge als Wert zur zusätzlichen Option `-F` (Langform `--display-format`) wie folgt an:

Individuelle Gestaltung des Ausgabeformats von aptitude bei der Suche nach dem Paket *tzdata*

```
aptitude search -F "%c%a | %15p" tzdata
i  | tzdata
v  | tzdata:i386
i  | tzdata-java
v  | tzdata-java:i386
v  | tzdata-jessie
v  | tzdata-jessie:i386
```

10.5 Für `aptitude` die Ausgabebreite festlegen

Ohne weitere Angaben benutzt `aptitude` zur Ausgabe die gesamte Breite des Terminals. Möchten Sie das auf einen bestimmten Wert festlegen, nutzen Sie dafür den Schalter `-w` (Langform `--width`) gefolgt von der Anzahl Zeichen. Das nachfolgende Beispiel zeigt Ihnen den Aufruf für eine Breite von 40 Zeichen. Überflüssige Zeichen schneidet `aptitude` in der Ausgabe ab.

Begrenzung der Ausgabe auf eine feste Breite

```
$ aptitude search -w 40 debtags
i   debtags      - Akiviert die Unterstü
p   debtags:i386 - Akiviert die Unterstü
p   python-debta - Vergleicht »hardware:
p   python3-debt - Vergleicht »hardware:
$
```

Dieser Wert korrespondiert mit der Einstellung `Aptitude::CmdLine::Package-Display-Width` in der Konfigurationsdatei zu `aptitude`.

10.6 Bei `aptitude` die Ausgabe sortieren

Möchten Sie die Ausgabe darüberhinaus noch sortieren, hilft Ihnen der Schalter `-O` (Langform `--sort`) weiter. Die Sortierung der Ausgabe erlaubt bspw. die Werte `installsize` (Installationsgröße), `name` (Paketname) und `version` (Versionsnummer).

Die Basiseinstellung ist `name,version`, wobei `aptitude` zuerst eine Sortierung anhand des Paketnamens und danach noch anhand der Paketversion durchführt. Somit erscheinen ältere Pakete in der Auflistung zuoberst.

Nachfolgend sehen Sie wiederum eine dreispaltige Ausgabe, die hier aus der Größe nach der Installation (Platzbedarf auf dem Speichermedium), dem Paketnamen sowie dem Namen und der EMailadresse des Paketmaintainers besteht. Zusätzlich ist die Ausgabe aufsteigend nach der Paketgröße sortiert. Recherchiert wird dabei nach allen Paketen, die im Namen die Zeichenkette `debtags` beinhalten.

Suche nach debtags-Paketen mit spezifischer Formatierung der Ausgabe und Sortierung

```
$ aptitude search -F "%I %5p,%m" --sort installsize debtags
79,9 kB  python-debtagshw      ,Enrico Zini <enrico@debian.org>
79,9 kB  python3-debtagshw     ,Enrico Zini <enrico@debian.org>
826 kB   debtags:i386          ,Enrico Zini <enrico@debian.org>
910 kB   debtags               ,Enrico Zini <enrico@debian.org>
$
```

10.7 `aptitude`-Gruppierung

10.7.1 Kommandozeile

Zur Gruppierung kennt `aptitude` den Schalter `--group-by`. Eine kurze Version des Schalters existiert u.E. bislang nicht. Als Wert sind die folgenden Möglichkeiten zulässig:

`archive`
: nach dem Enthaltensein eines Pakets in einer Veröffentlichung, bspw. `stable` oder `unstable`.

`auto`
: Gruppierung nach dem Paketnamen

`none`
: Darstellung aller Versionen in einer einzigen Liste ohne jegliche Sortierung

`package`
: Gruppierung nach dem Paketnamen

`source-package`
: Gruppierung nach dem Namen des Sourcepakets

`source-version`
: Gruppierung nach dem Namen und der Version des Sourcepakets

Diese Werte korrespondieren mit der Einstellung `Aptitude::CmdLine::Versions-Group-By` in der Konfigurationsdatei zu `aptitude`.

10.7.2 Textoberfläche

ToDo:

- Anordnung der Spalten in der Text-Modus-Oberfläche
- Breite der Spalten
- welche Spalten sind überhaupt darstellbar
- wie stelle ich das ein

10.8 `aptitude`-Farbschema anpassen

10.8.1 Standardvorgaben

- Standardfarben: siehe Beschreibung unter Abschnitt 6.3.2

10.8.2 Zwischen `aptitude`-Themes wechseln

- Theme: Farben und Anordnung
- siehe `aptitude`-Handbuch [aptitude-dokumentation-themes]
- zwei Themes werden mitgeliefert
  - `Dselect` (wie `dselect`) — ist das Standard-Theme
  - `Vertical-Split` — teilt die Darstellung senkrecht in Paketliste (links) und Beschreibung (rechts)
  - Konfigurationsdirektive: `Aptitude::Theme Vertical-Split;`

10.8.3 Eigene Farben vergeben

- für die einzelnen Strukturelemente eigene Farben festlegen
- siehe `aptitude`-Handbuch: Customizing text colors and styles [aptitude-dokumentation-text-colors-and-styles]
- Frage:
  - ist das empfehlenswert, oder stiftet das nicht eher Verwirrung?
  - Vorlieben und Gewohnheiten
  - Sehfähigkeiten (Farben, Kontrast)
  - Ausgabegerät, insbesondere Helligkeit

Kapitel 11

Mit `aptitude` Vormerkungen machen

Alle Paketoperationen, die wir Ihnen im Grundlagenteil ausführlich vorgestellt haben, wirken sich unmittelbar auf den aktuellen Paketbestand auf ihrem System aus. Das entspricht auch den üblichen Erwartungen im Alltag – Sie aktualisieren zuerst die lokale Liste der Pakete mit `apt-get update` oder `aptitude update` (siehe Abschnitt 8.39), wählen danach aus der Paketliste die Pakete aus, die hinzukommen, aktualisiert werden oder zu entfernen sind und führen anschließend die jeweilige Aktion mittels `aptitude install Paketname` respektive `aptitude full-upgrade Paketname` bzw. `aptitude remove Paketname` aus (siehe Abschnitt 8.36, Abschnitt 8.39 und Abschnitt 8.41).

`aptitude` kennt ein Konzept namens Vormerkungen. Es gestattet Ihnen, Paketoperationen zunächst Schritt für Schritt vorzubereiten und diese Vormerkungen zu einem späteren Zeitpunkt als Stapel auszuführen. Dazu gehören alle Aktionen, die den Paketbestand auf ihrem System verändern, wie bspw. die Installation, das Aktualisieren und das Entfernen von Paketen. `aptitude` merkt sich die einzelnen Aktionen und arbeitet diese ab, wenn Sie das Programm via `aptitude install` ohne weiteren Paketnamen aufrufen.

Vormerkungen mit Synaptic

Nicht verschweigen möchten wir Ihnen, dass Synaptic (siehe Abschnitt 6.4.1) ein ähnliches benanntes Konzept bietet. Unter dem Menüpunkt `Datei` verbergen sich die drei Einträge `Vorgemerkte Änderungen Speichern`, `Vorgemerkte Änderungen Einlesen` und `Vorgemerkte Änderungen Speichern unter`. Hier werden ihre Vormerkungen in einer von Ihnen festgelegten, lokalen Datei gespeichert, die Sie jederzeit wieder einlesen und durch Synaptic ausführen lassen können. Beachten Sie bitte, dass diese Vormerkungen in keinerlei Zusammenhang zu den Vormerkungen durch `aptitude` stehen.

11.1 Vormerkungen über die Kommandozeile durchführen

Dafür bietet Ihnen `aptitude` den Schalter `--schedule-only` an. Dieser Schalter ist gleichwertig zur Auswahl über die Textoberfläche und beliebig mit Vormerkungen daraus mischbar.

In nachfolgender Ausgabe sehen Sie, wie Sie die Vormerkungen zur Installation des Pakets *cssed*, zum Entfernen des Pakets *apt-doc* und der Aktualisierung des Pakets *iceweasel* samt dessen Abhängigkeiten *libmozjs24d* und *xulrunner-24.0* durchführen. Das abschließende Kommando *search* gibt Ihnen eine Übersicht zu den Paketoperationen, die sich `aptitude` nun gemerkt hat und welche Pakete zur Änderung anstehen (siehe auch Abschnitt 11.3).

Vormerkungen über die Kommandozeile durchführen

```
# aptitude --schedule-only install cssed
# aptitude --schedule-only remove apt-doc
# aptitude --schedule-only upgrade iceweasel
Auflösen der Abhängigkeiten ...
# aptitude search '!~akeep'
id  apt-doc                - Dokumentation für APT
pi  cssed                  - graphical CSS editor
```

```
iu  iceweasel                - Webbrowser auf Basis von Firefox
iu  libmozjs24d              - Mozilla SpiderMonkey JavaScript library
iu  xulrunner-24.0           - XUL + XPCOM application runner
#
```

Vormerkungen wieder aufheben
Möchten Sie die gewählten Vormerkungen nicht ausführen und stattdessen wieder rückgängig machen, heben Sie diese wieder auf. Die Details dazu entnehmen Sie dem Abschnitt 11.5.

11.2 Vormerkungen über die Textoberfläche durchführen

Um eine Paketoperation für eine spätere Verarbeitung vorzumerken, wählen Sie zunächst das gewünschte Paket aus der Paketliste aus. Tabelle 11.1 stellt die Tastenkombinationen zusammen, die Sie dafür benutzen können.

Tabelle 11.1: Tastenkombination für Vormerkungen in `aptitude`

Taste	Bedeutung
+	Paket installieren oder aktualisieren (*install* oder *upgrade*)
-	Paket entfernen (*remove*)
_	Paket vollständig entfernen (*purge*)
:	Paketversion behalten (*keep*)
=	Paketversion dauerhaft beibehalten (*hold*)
L	Paket nochmals installieren (*reinstall*)
U	alle aktualisierbaren Pakete zur Aktualisierung vormerken

Drücken Sie die Taste **g**, erhalten Sie danach zunächst nur eine Vorschau ihrer Vormerkungen (siehe Abschnitt 11.3). Drücken Sie die Taste **g** erneut, führt `aptitude` ihre Vormerkungen auch tatsächlich aus (siehe Abschnitt 11.6).

Ihre bereits gewählten Vormerkungen können Sie jederzeit wieder aufheben. Die Details dazu entnehmen Sie Abschnitt 11.5.

11.3 Bestehende Vormerkungen anzeigen

`aptitude` kennt zwei Wege, um Ihnen diese Informationen anzuzeigen – einerseits über die Kommandozeile und andererseits über die Textoberfläche. Nachfolgend gehen wir davon aus, dass Sie die gewünschten Aktionen bereits vorbereitet haben (siehe dazu Abschnitt 11.1 und Abschnitt 11.2).

Über die *Kommandozeile* ist `aptitude` recht auskunftsfreudig. Dazu benutzen Sie das Unterkommando `search` mit der Option `~akategorie` oder als Langform `?action(kategorie)`. Als Wert für die Kategorie können Sie eines der folgenden Werte angeben:

`install`
listet alle Pakete auf, die installiert werden (siehe Abschnitt 8.36)

`upgrade`
listet alle Pakete auf, die durch eine neuere Version ersetzt werden (siehe Abschnitt 8.39)

`downgrade`
listet alle Pakete auf, die durch eine ältere Version ersetzt werden (siehe Abschnitt 8.40)

`remove`
listet alle Pakete auf, die gelöscht werden (siehe Abschnitt 8.41)

purge
listet alle Pakete auf, die vollständig gelöscht werden (siehe Abschnitt 8.41)

hold
listet alle Pakete auf, deren Version explizit beibehalten wird (siehe Abschnitt 2.15)

keep
listet alle Pakete auf, die automatisch beibehalten werden (siehe Abschnitt 2.15)

Die nachfolgende Ausgabe ist das Äquivalent zu Abbildung 11.1 im Terminal. Bitte beachten Sie dabei, dass Sie die zusätzliche `aptitude`-Option mit der Kategorie in Anführungszeichen einschließen, damit die ausführende Shell diese Option nicht interpretiert und ggf. verändert.

Ausgabe der vorgemerkten Paketoperationen über die Kommandozeile

```
# aptitude search '~ainstall'
pi  cssed                          - graphical CSS editor
# aptitude search '~aremove'
id  apt-doc                        - Dokumentation für APT
# aptitude search '~aupgrade'
iu  iceweasel                      - Webbrowser auf Basis von Firefox
iu  libmozjs24d                    - Mozilla SpiderMonkey JavaScript library
iu  xulrunner-24.0                 - XUL + XPCOM application runner
#
```

`aptitude` kann Ihnen auch berichten, welche Pakete sich verändern und nicht in dem bestehenden Zustand gehalten werden. Dabei hilft Ihnen die Option `!~akeep` zum Unterkommando `search`. Die Liste der Pakete ist deckungsgleich mit dem Ergebnis aus obiger Liste. In der linken Spalte der Ausgabe sehen Sie den Paketstatus, gefolgt vom Paketnamen und der Kurzbeschreibung des Pakets in der rechten Spalte.

Pakete darstellen, die nicht in dem bestehenden Zustand gehalten werden

```
# aptitude search '!~akeep'
id  apt-doc              - Dokumentation für APT
pi  cssed                - graphical CSS editor
iu  iceweasel            - Webbrowser auf Basis von Firefox
iu  libmozjs24d          - Mozilla SpiderMonkey JavaScript library
iu  xulrunner-24.0       - XUL + XPCOM application runner
#
```

Darstellung der Pakete, die aktualisiert werden können
Um herauszufinden, welche weiteren Pakete aktualisierbar wären, lesen Sie das Vorgehen unter Aktualisierbare Pakete anzeigen in Abschnitt 8.12 nach.

In der *Textoberfläche* drücken Sie hingegen die Taste **g**. Daraufhin sehen Sie eine Darstellung ähnlich zu Abbildung 11.1, in der die einzelnen Paketoperationen gruppiert sind. Als Kategorien bestehen derzeit:

- Pakete, die automatisch in ihrem derzeitigen Zustand gehalten werden (siehe Abschnitt 2.15),
- Pakete, die installiert werden (siehe Abschnitt 8.36),
- Pakete, die zurückgehalten werden (siehe Abschnitt 2.15),
- Pakete, die entfernt werden (siehe Abschnitt 8.41) und
- Pakete, die aktualisiert werden (siehe Abschnitt 8.39).

`aptitude` zeigt Ihnen nur die Kategorien an, in denen überhaupt Paketoperationen stattfinden. Alle anderen Kategorien werden von vornherein ausgeblendet. Im vorliegenden Fall ist nur das Paket *cssed* zur Installation vorgemerkt, *apt-doc* wird hingegen entfernt und *iceweasel* von der Version `24.8.0esr-1~deb7u1` auf `24.8.1esr-1~deb7u1` aktualisiert. Zwei weitere Pakete werden ebenfalls aktualisiert, sind aber in der Auflistung nicht sichtbar.

Jede Paketoperation wird gesondert farblich hervorgehoben, damit Ihnen auch optisch deutlich wird, was mit den ausgewählten Paketen passieren wird. Mehr zur Kennzeichnung durch die verschiedene Farben lesen Sie in Abschnitt 6.3.2 und Abschnitt 10.8.

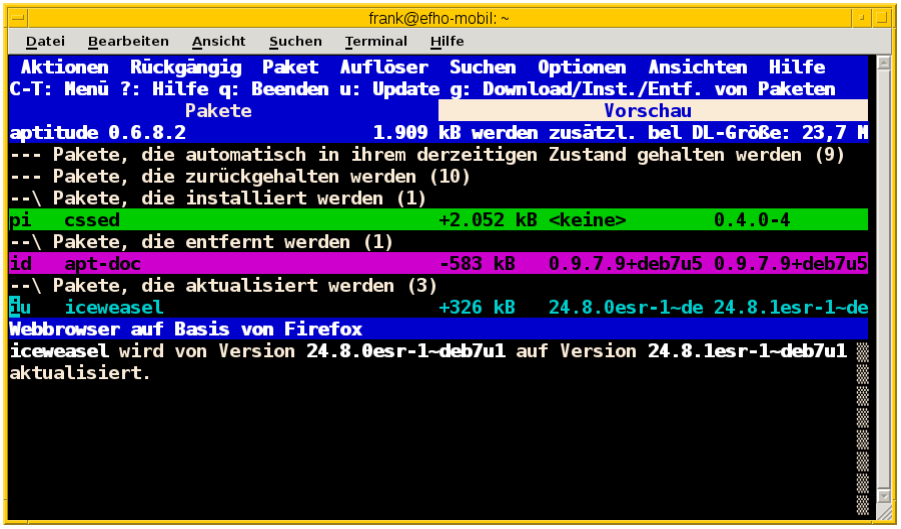

Abbildung 11.1: Paketoperationen anzeigen, die zur Ausführung anstehen

Änderungen der Vormerkungen
In der Vorschau können Sie nochmals die vorgemerkten Paketoperationen verändern. Die Ansicht wird dabei aber nicht automatisch neu aufgebaut.

11.4 Vormerkungen simulieren

Insbesondere bei vielen Vormerkungen oder wenn Sie grundlegende Pakete ändern, geht mitunter die Übersicht verloren, welche Pakete in der Gesamtheit überhaupt betroffen sind. Um vorher auszuprobieren, was passieren wird, wenn Ihre Vormerkungen durch `aptitude` ausgeführt werden, bietet Ihnen das Programm daher die entsprechende Option `-s` (Langform `--simulate`) an. Die nachfolgende Ausgabe zeigt das Ergebnis der Simulation für die vorgemerkten Paketoperationen analog zu Abbildung 11.1 in Abschnitt 11.3.

Zukünftige Aktionen auflisten durch Simulation

```
# aptitude install -s
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  cssed
Die folgenden Pakete werden ENTFERNT:
  apt-doc
Die folgenden Pakete werden aktualisiert:
  iceweasel libmozjs24d xulrunner-24.0
3 Pakete aktualisiert, 1 zusätzlich installiert, 1 werden entfernt und 19 nicht  ↩
    aktualisiert.
22,9 MB/23,7 MB an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 1.909  ↩
    kB zusätzlich belegt sein.
Möchten Sie fortsetzen? [Y/n/?]
Pakete würden heruntergeladen/installiert/entfernt werden.
#
```

Automatisches Ausführen
Kombinieren Sie obigen `aptitude`-Aufruf zur Simulation mit dem Parameter `-y` (Langform `--assume-yes`), entfällt die manuelle Beantwortung der Frage „Möchten Sie fortsetzen?“. In diesem Fall werden alle Fragen automatisch mit „Ja“ beantwortet.

11.5 Vormerkungen wieder aufheben

Natürlich bietet Ihnen `aptitude` auch die Möglichkeit, die bereits bestehenden Vormerkungen wieder aufzuheben. Für die *Kommandozeile* verfügt `aptitude` über ein Unterkommando namens `keep-all`. Sie rufen es ohne weitere Optionen auf, wie Ihnen die nachfolgende Ausgabe zeigt.

Aufheben der Vormerkungen

```
# aptitude keep-all
Es werden keine Pakete installiert, aktualisiert oder entfernt.
0 Pakete aktualisiert, 0 zusätzlich installiert, 0 werden entfernt und 22 nicht  ↩
    aktualisiert.
0 B an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 0 B zusätzlich  ↩
    belegt sein.

#
```

In der *Textoberfläche* wählen Sie stattdessen den äquivalenten Menüpunkt `Aktionen` → `Noch ausstehende Aktionen abbrechen` aus (siehe Abbildung 11.2).

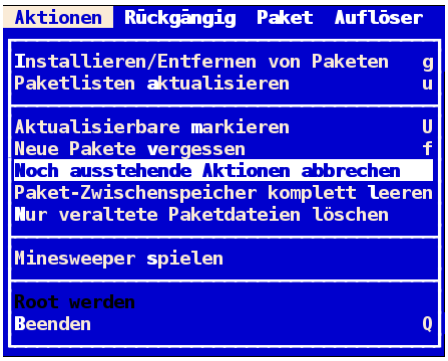

Abbildung 11.2: Vormerkungen abbrechen

11.6 Vormerkungen ausführen

Nachdem Sie die gewünschten Paketoperationen zusammengestellt, vorgemerkt und überprüft haben, fehlt noch der abschließende Schritt – die Umsetzung. In der *Kommandozeile* genügt es vollkommen, wenn Sie `aptitude` mit dem Unterkommando `install` ohne weitere Paketnamen aufrufen. Zusätzliche Parameter sind in diesem Fall nicht erforderlich.

```
# aptitude install
Die folgenden NEUEN Pakete werden zusätzlich installiert:
  cssed
Die folgenden Pakete werden ENTFERNT:
  apt-doc
Die folgenden Pakete werden aktualisiert:
  iceweasel libmozjs24d xulrunner-24.0
3 Pakete aktualisiert, 1 zusätzlich installiert, 1 werden entfernt und 19 nicht  ↩
    aktualisiert.
```

```
22,9 MB/23,7 MB an Archiven müssen heruntergeladen werden. Nach dem Entpacken werden 1.909 ↩
    kB zusätzlich belegt sein.
Möchten Sie fortsetzen? [Y/n/?] y
Holen: 1 http://security.debian.org/ wheezy/updates/main iceweasel i386 24.8.1esr-1~deb7u1 ↩
    [2.873 kB]
Holen: 2 http://security.debian.org/ wheezy/updates/main xulrunner-24.0 i386 24.8.1esr-1~ ↩
    deb7u1 [18,4 MB]
69% [2 xulrunner-24.0 12,9 MB/18,4 MB 70%]                          699 kB/s 10 s
...
Aktueller Status: 19 aktualisierbare Pakete [-3].
#
```

In der *Textoberfläche* erfolgt das mit der Taste **g**. Daraufhin arbeitet `aptitude` ihre vorbereitete Liste der Vormerkungen ab.

11.7 Risiken und Seiteneffekte

Die Idee, die `aptitude` mit dem Konzept der Vormerkungen transportiert, ist toll und ungemein praktisch. Es erleichtert komplexe Veränderungen des Paketbestands auf Ihrem Debian-System. Es hilft Ihnen dabei, den ganzen Arbeitsberg in handhabbare Einzelschritte zu zerlegen und später „alles in einem Rutsch“ ablaufen zu lassen.

Sind Sie jedoch mit dem Konzept noch weniger vertraut, lauern kleine Fallen, die durchaus für Überraschungen sorgen können.

1. `aptitude` merkt sich, welche Vormerkungen Sie über die *Textoberfläche* vorgenommen haben. Beenden Sie `aptitude` mittels **q**, speichert es die Vormerkungen. Brechen Sie `aptitude` hingegen mit Ctrl-c ab, werden die Vormerkungen nicht aktualisiert und bleiben so, wie sie bisher sind.

2. Nutzen Sie `aptitude` eher selten, ist nicht auszuschließen, dass Sie die vorher gemerkten Aktionen inzwischen nicht mehr präsent haben. In Folge können Änderungen des Paketbestands passieren, die Sie (nicht mehr) zuordnen können. Wir raten Ihnen daher, vorher die eventuell bereits bestehenden Vormerkungen zu prüfen (siehe Abschnitt 11.3) und erst danach weitere Änderungen im Paketbestand zu veranlassen.

3. Wechseln Sie in der Benutzung zwischen `aptitude` und anderen Paketverwaltungsprogrammen hin und her, wird es auch sehr spannend. APT weiß bspw. nichts von den Vormerkungen seitens `aptitude` und kann diese daher auch nicht berücksichtigen. Verwirrung auf allen Seiten ist hier zu erwarten und gegebenenfalls werden andere Paketoperationen ausgeführt, als sie beabsichtigt haben (siehe dazu auch Kapitel 12).

Kapitel 12

APT und `aptitude` mischen

12.1 Hintergrund

Immer wieder taucht die Frage auf, ob APT und `aptitude` identisch sind oder sich beide Werkzeuge im Alltag miteinander kombinieren lassen. Aus unserer Sicht sollte Ihnen als Leser bisher mehr als deutlich geworden sein, dass zwar beide Werkzeuge das gleiche Ziel verfolgen, jedoch etwas anders „ticken“. Zu klären ist daher, ob sich beide Werkzeuge bei deren gemischter Verwendung wechselseitig ins Gehege kommen und welche Situationen unkritisch sind.

Für viele Nutzer stellt sich die Frage nicht, weil die Präferenz für ein bestimmtes Programm seit längerem feststeht und dieses aus purer Gewohnheit für die Erledigung aller Aufgaben im Kontext der Paketverwaltung verwendet wird — egal, wie umständlich das auch ist. Müssen Sie Beispielanleitungen oder auch HowTos nachvollziehen, wird es jedoch sehr spannend. Da besteht keinerlei Einheitlichkeit und zeitweise wird APT beschrieben, ein andermal `aptitude` genutzt. Da blindes Vertrauen nie gut ist, sollten Sie einschätzen können, was passiert, wenn Sie der Anleitung folgen und „den anderen“ Paketmanager verwenden. Es hilft Ihnen auch dabei, zu wissen, wie Sie die beschriebenen Aktionen in die Handlungsschritte für den Paketmanager übersetzen, den Sie bevorzugen.

Darüber nachdenken führt dazu, dass Sie Ihren Arbeitsfluss Revue passieren lassen und die Handhabung der Programme im Alltag hinterfragen. Dabei können sich Gewohnheiten verändern und Blickwinkel erweitern. Eine Antwort auf die Fragen „Welches Programm erleichtert mir die Aufgaben? Was kann es besonders gut, was ist praktikabel und was nicht?“ eröffnet sich.

12.2 Sollten Sie das überhaupt machen?

An diesem Punkt sind Pro und Contra genau abzuwägen. Diese Abschätzung fällt natürlich leichter, umso mehr Sie mit den einzelnen Werkzeugen zur Paketverwaltung vertraut sind.

Dafür spricht die grundlegende Philosophie, auf der UNIX/Linux-Systeme basieren. Setzen Sie ein Programm stets genau für die Aufgabe(n) ein, für die es am besten passt. Zudem hat jedes Werkzeug Eigenschaften, mit denen es sich von anderen abhebt und womit Sie bestimmte Aufgaben besonders schnell oder möglichst gut erledigen können.

Dagegen spricht, dass die Programme in kleinen Details verschieden sind, auch wenn sie in der Gesamtheit das gleiche Ergebnis liefern. Ins Gewicht fällt hierbei insbesondere die Synchronisation der Informationen bzgl. Status und Vormerkungen zu den Softwarepaketen zwischen `dpkg`, APT und `aptitude`, was bislang nicht vollständig erfolgt. Damit Sie zwischen den verschiedenen Programmen wechseln können, muss jedes wissen, was das andere macht bzw. gemacht hat, oder eben nicht macht. Das ist derzeit noch nicht gegeben und wächst erst Stück für Stück zusammen.

Und nun? Mischen gelingt Ihnen sorgenfrei, wenn Sie wissen, was Sie tun und wie die einzelnen Werkzeuge zusammenspielen. Für den Zusammenhang zwischen `dpkg`, APT, `aptitude` empfehlen wir Ihnen das Kapitel Zusammenspiel von dpkg und APT in Abschnitt 2.5. Sind Sie sich diesbezüglich noch unsicher, kombinieren Sie am besten zunächst nur die Aktionen, die ungefährlich sind und lesen die benötigten Details zu den beteiligten Werkzeugen nach, bevor Sie diese kreuz und quer einsetzen.

12.3 Was ist zu beachten, wenn Sie das machen

Unkritisch sind in jedem Fall alle Paketoperationen, bei denen Sie nur *lesend* auf den Paketbestand zugreifen, d.h. es in diesem nicht zu einer Veränderung kommt. Dazu zählen z.B. das Erfragen des Paketstatus (siehe Abschnitt 8.4), die Liste der installierten Pakete anzeigen und deuten (siehe Abschnitt 8.5), die neuen Pakete anzeigen (siehe Abschnitt 8.8), die Pakete nach Prioritäten finden (siehe Abschnitt 8.9), die Installationsgröße eines Pakets bestimmen (siehe Abschnitt 8.13), die Paketabhängigkeiten anzeigen (siehe Abschnitt 8.17), die Herkunft der Pakete klären (siehe Abschnitt 8.18) und über den Paketinhalt suchen (siehe dazu Abschnitt 8.22 und Abschnitt 8.23).

Definitiv als bedenklich schätzen wir ein, wenn Sie APT und `aptitude` im fliegenden Wechsel für alle Paketoperationen benutzen, bei denen der Paketbestand *verändert wird* oder entsprechende Vormerkungen dazu getroffen werden. Weiterhin treten Seiteneffekte auf, wenn mehrere Programme zur Paketverwaltung gleichzeitig geöffnet sind, bspw. `aptitude`, Synaptic und SmartPM. Jedes der genannten Werkzeuge versucht, für die einzelnen Aktionen die Paketdatenbank exklusiv nutzen. Funken an dieser Stelle andere Programme dazwischen, entstehen Konflikte mit unvorhersagbarem Ergebnis.

Um Letzteres zu verhindern, raten wir Ihnen zur konsequenten Benutzung des gleichen Werkzeugs. Zu den Operationen zählen z.B. das Beziehen und Installieren eines Pakets (siehe Abschnitt 8.31, Abschnitt 8.32 und Abschnitt 8.36) sowie das Ändern der Paketversion (siehe Abschnitt 8.39 und Abschnitt 8.40) und Deinstallieren bestehender Pakete (siehe Abschnitt 8.41 und Abschnitt 8.42).

Im Alltag hat sich besipielsweise bewährt, dass Sie zunächst über die Textoberfläche von `aptitude` oder die Debian-Webseite nach dem passenden Paket suchen. In Folge installieren Sie die konkreten, gewünschten Pakete via `apt-get` oder `apt` nach. Damit kombinieren Sie eine graphische bzw. textbasierte Oberfläche mit der unmißverständlichen Direktheit einer Kommandozeile.

12.4 Empfehlungen für Dokumentation und Beispiele

Stöbern Sie nach Beispielen zur Paketverwaltung oder auch in der Dokumentation zu komplexeren Softwarepaketen, wird der stete Wechsel und die Vermischung von `dpkg`, APT und `aptitude` offenkundig. Bislang hat sich dazu noch kein Standard durchgesetzt, welches der vorgenannten Werkzeuge zur Installation genutzt oder empfohlen wird. Jeder Entwickler und Autor folgt an dieser Stelle seinen eigenen Präferenzen.

Verfassen Sie selbst Dokumentation, Beispiele oder Anleitungen, raten wir Ihnen zu folgendem Vorgehen:

- geben Sie die verwendeten Programme stets in identischer Art und Weise an. Das betrifft insbesondere die Groß- und Kleinschreibung bzgl. APT und `apt`, da sonst Gefahr zur Verwirrung besteht, welches Werkzeug tatsächlich gemeint ist.
- geben Sie bei den Optionen und Schaltern sowohl die genutzte Kurz- als auch die Langversion an, sofern diese existieren und bekannt sind. Erklären Sie zusätzlich, warum Sie die von Ihnen verwendeten Optionen nutzen und diese in der genannten Reihenfolge Verwendung finden.

Mit diesen Schritten steigt das Verständnis des Kommandoaufrufs und vereinfacht es nicht nur Einsteigern, vorab zu verstehen, was da passieren soll. Der vollständige Aufruf mit Beschreibung klärt Missverständnisse und hilft Ihnen auch dabei, Fehler zu vermeiden.

Bei der Benennung und Auswahl der Werkzeuge spielen Gewohnheit und die Faulheit beim Tippen des Aufrufs eine Rolle. Nutzen Sie einen expliziten Aufruf, der exakt so funktioniert und nicht anders, hilft neben einem Hinweis auch ein wenig Hintergrundinformation zum Aufruf. Dazu zählen insbesondere Seiteneffekte, von denen Sie wissen und die Sie durch die spezifischen Parameter jedoch vermeiden möchten. Ein Verweis auf Alternativen und zusätzliche Dokumentation rundet den Text ab.

Kapitel 13

Erweiterte Paketklassifikation mit Debtags

13.1 Einführung

Wie bereits in der Einführung zum Buch in Teil I deutlich wurde, umfasst die Klassifikation der Pakete in Debian unterschiedliche Stufen. Neben der Verfügbarkeit verschiedener Veröffentlichungen (siehe Abschnitt 2.10) erfolgt eine Paketzuordnung anhand der Distributionsbereiche (siehe Abschnitt 2.9) und lediglich *einer möglichen* Softwarekategorie (siehe Abschnitt 2.8). Ein Paket können Sie auch auf der Grundlage des Paketnamens selektieren, sofern Sie sich mit dieser etwas doch recht eigenen Logik vertraut fühlen.

Obwohl sich diese Vorgehensweisen über die letzten 20 Jahre bewährt haben, ergeben sich daraus mittlerweile eine ganze Reihe von Problemen. Diese rühren schlicht und einfach aus der schieren Anzahl an Paketen, die inzwischen erfreulicherweise für Debian zur Verfügung stehen.

- Der Paketüberblick geht verloren und die Auswahl und das Finden eines bestimmten Pakets gerät mehr oder weniger zum zufälligen Ereignis.
- Das Klassifikationsraster zur Einordnung der Pakete in die bestehenden Softwarekategorien ist zu grob und lässt nur einen einzigen, vorher bestimmten Blickwinkel zu. Der Maintainer eines Pakets muss daher genau abwägen, welche Paketkategorie überwiegt oder am besten passt und trägt diese Kategorie in der `control`-Datei des Debian-Pakets ein (siehe Aufbau eines Debian-Pakets in Abschnitt 4.2.3).
- Es ist keine Mehrfachzuordnung möglich, wenn ein Programm verschiedene Aspekte umfasst und thematisch in unterschiedliche Kategorien passt.
- Die Suche mittels APT und `aptitude` gelingt nur in der korrekten Schreibweise über den Paketnamen bzw. einem Fragment daraus, alternativ über ein Muster oder einen Begriff aus der Paketbeschreibung. `aptitude` gestattet Ihnen zwar dazu auch die Verwendung Regulärer Ausdrücke, setzt aber den gekonnten Umgang damit voraus. Eine Recherche nach der thematischen Ähnlichkeit, einer konkreten Eigenschaft des Pakets oder dem Funktionsumfang ist nicht möglich. Ausführlich besprechen wir diese Recherchemöglichkeiten bereits unter Pakete über den Namen finden in Abschnitt 8.19.

Gelingt Ihnen die Recherche über die Paketkategorien oder den Namen des Pakets nicht, geht das Paket in der Masse der Möglichkeiten unter und bleibt letztendlich unentdeckt. Im Ergebnis führt das vor allem dazu, dass Sie als Debian-Benutzer mehr und mehr Experte sein müssen, um sich innerhalb der Debian-Paketliste zurechtzufinden. Für die Praxis heißt das, dass Sie ungefähr wissen müssen, wo das betreffende Paket derzeit in der Hierarchie eingeordnet wurde. Das Gefühl dafür erlangen Sie meist erst im Laufe der Zeit. So toll es auch ist, dass Debian so vielfältig bezüglich seiner Pakete ist, wird es doch zunehmend anspruchsvoller, sich den Überblick über die Komponenten zu erarbeiten und diesen zu behalten.

13.2 Kurzinfo zum Debtags-Projekt

Um den eingangs zunehmend stärker ins Gewicht fallenden Widrigkeiten zu begegnen, nahm sich der italienische Debian-Entwickler Enrico Zini im Jahre 2003 des Problems an. Er orientierte sich dabei an der Art und Weise, wie Bücher und Dokumente von je her anhand von Schlüsselworten und Kategorien indexiert und klassifiziert werden. Zur Debian-Entwicklerkonferenz

DebConf im Jahre 2005 in Helsinki [DebConf5] stellte er sein Projekt *Debtags* [Debian-Debtags-Old] vor, welches die Debianpakete und deren Beschreibung um geeignete Stichworte oder Attribute ergänzt und damit die Granularität der Suche deutlich erhöht. Seit Anfang 2016 ist die Webseite des Debtags-Projekts Bestandteil der offiziellen Webseite des Debian-Projekts [Debian-Debtags] (siehe auch „Debtags Webseite“ Abschnitt 13.3).

Der Projektname *Debtags* leitet sich von den beiden Worten *Debian* und *tags* ab, wobei sich letzteres mit Schlagwort, Markierung, Stichwort, Etikettierung oder Attribut ins Deutsche übersetzen lässt. Enrico Zini pflegt dazu das gleichnamige Paket *debtags* [Debian-Paket-debtags], welches sehr schnell in den regulären Paketbestand aufgenommen wurde. Seit Debian 4.0 *Etch* sind die Debtags ein regulärer Bestandteil jeder Paketbeschreibung.

Den hier angestoßenen Vorgang kennen Bibliothekare sowie Spezialisten zum Information Retrieval und zur Suchmaschinenoptimierung – engl. SEO als Abkürzung für Search Engine Optimization – unter dem Fachbegriff Verschlagwortung von Inhalten und Indexierung von Dokumenten. Gleiche Sachverhalte werden dabei durch einheitliche Begriffe repräsentiert. In dem hier genutzten Klassifikationsschema entspricht ein verwendeter Begriff einem Aspekt oder einer spezifischen Eigenschaft eines Pakets.

Jedes Paket kann beliebig viele Schlagworte besitzen, sogenannte Mehrfachattribute oder Facetten. Daher heißt das Klassifikationsschema auch Facettenklassifikation – englisch *faceted classification*. Die im Schema verwendeten Begriffe orientieren sich an der Umgangssprache und umfassen bspw. *interface* (Benutzerschnittstelle des Programms), *protocol* (verwendetes oder unterstütztes Netzwerkprotokoll) oder *works-with-format* (unterstütztes bzw. akzeptiertes Datenformat). Sowohl die Liste der verwendeten Schlagworte, als auch die bereits vergebenen Schlagworte für ein Paket können bei Bedarf von jedem Interessierten unkompliziert ergänzt und korrigiert werden.

Als Datenbank zu den Schlagworten fungiert eine Textdatei, die sich unter `/var/lib/debtags/vocabulary` befindet. Neben dem Hauptbegriff („Facette“) finden Sie eine Beschreibung und ggf. auch eine Statusinformation. Nachfolgend sehen Sie einen Ausschnitt aus dem Abschnitt `interface`.

Klassifikation der Benutzerschnittstelle über die Facette interface (Ausschnitt)

```
Facet: interface
Description: User Interface
 What kind of user interface the package provides
Status: needing-review

Tag: interface::3d
Description: Three-Dimensional

Tag: interface::commandline
Description: Command Line

Tag: interface::text-mode
Description: Text-based Interactive
```

Wurden die Debian-Pakete entsprechend markiert, vereinfacht sich darüber die Suche nach passenden Paketen erheblich (siehe Vergebene Schlagworte anzeigen in Abschnitt 13.5 und Suche anhand der Schlagworte in Abschnitt 13.6). Damit profitieren Sie als Nutzer sofort von dem Klassifikationsschema und können bei Interesse gleichzeitig zu dessen Komplettierung beitragen, indem Sie Debianpakete, die noch unvollständig kategorisiert oder gänzlich ohne Attribute sind, um weitere Schlagworte ergänzen. In Folge haben alle Debian-Benutzer kurzfristig einen spürbaren Vorteil davon.

Innerhalb kürzester Zeit erfolgte zudem eine nahtlose Integration von Debtags in die bestehenden Paketmanager, sodass das Feature allgemein verfügbar wurde. Leider ist diese nützliche Eigenschaft bislang nicht allen Benutzern präsent. Wir erhoffen uns einen höheren Verbreitungs- und Nutzungsgrad, indem wir nachfolgend das Projekt und insbesondere dessen Werkzeuge ausführlicher beleuchten.

13.3 Webseite zum Projekt

Koordiniert wird Debtags über die Webseite zum Projekt, bis zu Ende 2015 erreichbar unter der historischen Webpräsenz [Debian-Debtags-Old]. Zu Beginn des Jahres 2016 hat Enrico Zini aufgrund geänderter Prioritäten seine Aktivität im Debtags-Projekt eingeschränkt. Die bestehenden webbasierten Werkzeuge wurden daraufhin in die offizielle Webseite des Debian-Projekts integriert [Debian-Debtags]. Um ihrerseits am Projekt mitzuwirken, ist von nun an ein Debian-Account erforderlich.

Auf der Projektseite finden Sie neben den Basisdaten eine ausführliche Dokumentation inklusive der Informationen zum Application Programming Interface (API) sowie zum genutzten Vokabular, d.h. den verwendeten Schlagworten. Weiterhin gehört auch eine statistische Auswertung dazu, um die Menge und die Verteilung der genutzten Schlagworte nachvollziehen zu können [Debian-Debtags-Statistics].

Über vorbereitete Formulare recherchieren Sie sowohl paketbezogen als auch anhand der Schlagworte. Für jedes Paket werden Ihnen diese angezeigt. Im *Debtags Editor* können diese von Ihnen ergänzt und modifiziert werden (siehe Abbildung 13.10). Alle darüber vorgenommenen Änderungen fließen in das genutzte Vokabular und die Datenbank zur Verwaltung und Speicherung der Schlagworte ein und werden sofort wirksam. Bei einer späteren Veröffentlichung des *debtags*-Pakets sind dann auch Ihre Beiträge ganz offiziell im aktualisierten Paket enthalten und somit für alle Benutzer verfügbar.

Die Webseite ist hervorragend konzipiert und bildet im Vergleich zu den später besprochenen graphischen Programmen eine recht einfach zu bedienende Schnittstelle zu den cleveren Werkzeugen aus dem *debtags*-Paket. Diese Werkzeuge stellen wir Ihnen in Abschnitt 13.4 genauer vor.

13.4 Debtags-Werkzeuge

Mittlerweile hat das Debtags-Konzept entsprechende Verbreitung gefunden und ist in einer ganzen Reihe von Werkzeugen verfügbar. Der Funktionsumfang variiert dabei erheblich.

Das Herzstück auf der *Kommandozeile* bildet das Paket *debtags* [Debian-Paket-debtags]. Dieses beinhaltet die Programme `debtags`, `debtags-fetch` und `debtags-submit-patch`. Ersteres zeigt Ihnen die bereits vergebenen Schlagworte für ein Paket an und ermöglicht Ihnen anhand der Schlagworte in der Paketdatenbank eine Suche (siehe dazu Abschnitt 13.6). Mit den anderen beiden Programmen stöbern Sie im gesamten Vokabular („Schlagwortschatz“), nehmen darin Veränderungen vor und laden ihre Änderungen zur zentralen Vokabulardatenbank hoch (siehe Abschnitt 13.8).

Ebenso unverzichtbar ist das Paket *dctrl-tools* [Debian-Paket-dctrl-tools], welches Ihnen die Recherche im Paketbestand erleichtert und dabei die vergebenen Schlagworte der Pakete auswertet. Es stellt mehrere Programme bereit, die jedoch stets als symbolische Links auf das Kommandozeilenwerkzeug `grep-dctrl` ausgeführt sind und dieses mit spezifischen Parametern aufrufen. Dazu zählen `grep-available`, `grep-aptavail`, `grep-debtags` und `grep-status`. Anwendungsbeispiele zu `grep-available` sind das Auflisten bekannter Paketnamen (siehe Abschnitt 8.3) sowie das Finden von Paketen anhand der Begriffe, die in der Paketbeschreibung enthalten sind (siehe dazu Abschnitt 8.20). Während Sie mittels `grep-aptavail` lediglich in der Liste der verfügbaren Pakete und über den Namen der darin enthaltenen Dateien stöbern (siehe ebenfalls Abschnitt 8.3 und Abschnitt 8.23), benutzt `grep-debtags` stattdessen die vergebenen Debtags als Grundlage zur Recherche. Mit dem Werkzeug `grep-status` erfragen Sie hingegen den aktuellen Status eines Pakets (siehe dazu mehr in Abschnitt 8.4).

Eine kleine und zunächst unscheinbar wirkende Anwendung ist `ara` aus dem gleichnamigen Debianpaket [Debian-Paket-ara]. Ebenso wie das vorgenannte `grep-dctrl` stöbern Sie damit im Paketbestand. Mittels boolescher Ausdrücke kombinieren Sie die gewünschten Suchfelder. Die nachfolgende Ausgabe präsentiert einen Ausschnitt des Suchergebnisses in tabellarischer Form bestehend aus dem Paketnamen, dessen Größe und dem Paketmaintainer beispielhaft zu allen Debianpaketen, die zur Kategorie *utils* (Werkzeuge) gehören, zudem vom Window Manager XFCE abhängen oder kleiner als 10000 Bytes sind und bzgl. der Priorität als *optional* eingestuft sind (siehe dazu Abschnitt 2.13).

Recherche mittels ara nach optionalen Paketen zum Window Manager XFCE (Auswahl)

```
$ ara -fields Package,Size,Maintainer:30 -table 'section=utils & (depends:(xfce) | size ↩
    <10000) & priority=optional'
+--------------------------------+------+--------------------------------+
| Package                        | Size | Maintainer                     |
+--------------------------------+------+--------------------------------+
| acpitail                       | 8340 | Debian Acpi Team <pkg-acpi-... |
| athena-jot                     | 9876 | Francesco Paolo Lovergine <... |
| autotrash                      | 9796 | Lorenzo De Liso <blackz@ubu... |
| binclock                       | 9540 | Nico Golde <nion@debian.org>   |
| colortest-python               | 9052 | Jari Aalto <jari.aalto@cant... |
| createfp                       | 9982 | Rene Engelhard <rene@debian... |
| ddir                           | 9776 | Jari Aalto <jari.aalto@cant... |
| eatmydata                      | 7778 | Modestas Vainius <modax@deb... |
| ksshaskpass                    | 9426 | Armin Berres <armin+debian@... |
| laptop-detect                  | 5212 | Otavio Salvador <otavio@deb... |
```

```
| leave                          | 7584 | Josip Rodin <joy-packages@d... |
| zeitgeist                      | 7570 | Siegfried-Angel Gevatter Pu... |
| zinnia-utils                   | 5336 | IME Packaging Team <pkg-ime... |
| ziptorrent                     | 7714 | Fathi Boudra <fabo@debian.org> |
+--------------------------------+------+--------------------------------+
$
```

Die in Abschnitt 6.4 bereits vorgestellten *graphischen Werkzeuge* Synaptic, SmartPM, das Ubuntu Software Center, Gdebi und PackageKit können bislang nicht mit dem Thema Debtags umgehen. Stattdessen entstanden auf der Grundlage der Bibliotheken zu Debtags mehrere Alternativen, die von Ihnen zum Teil ein sehr unübliches Bedienritual erfordern. Dazu gehören Package-Search [Debian-Paket-packagesearch], Adept [Debian-Paket-adept] und Xara [Debian-Paket-xara-gtk]. Letzteres ist die X11-Version des bereits oben gezeigten Kommandozeilenwerkzeugs `ara`. Alle Programme bieten die Suche anhand der Debtags über die Paketdatenbank an und verfügen darüberhinaus auch über eine einfache Schnittstelle zur Paketverwaltung.

Im Paket *goplay* [Debian-Paket-goplay] verbergen sich die einzelnen Werkzeuge `goadmin`, `golearn`, `gonet`, `gooffice`, `goplay`, `gosafe`, `goscience` und `goweb`. Jedes der genannten Programme ist auf eine spezifische Paketkategorie von Debian ausgerichtet, so z.B. `goplay` auf Spiele (siehe Abbildung 13.1), `golearn` auf Lernprogramme und `goscience` auf wissenschaftliche Werkzeuge (siehe dazu Abschnitt 2.8).

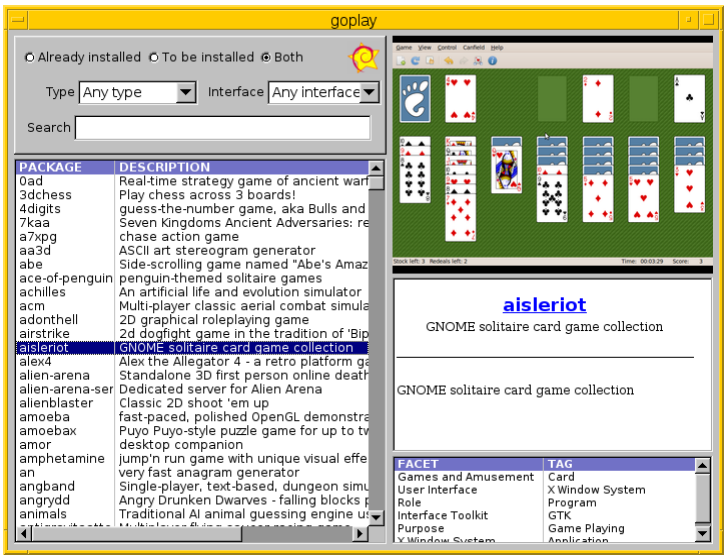

Abbildung 13.1: Suche nach Spielen anhand von Debtags

Zur vereinfachten Bearbeitung des Vokabulars hat Enrico Zini vor Jahren eine graphische Benutzeroberfläche namens *Debtags Editor* entworfen (Paket *debtags-edit* [Debian-Paket-debtags-edit]. Diese befindet sich derzeit im Dornröschenschlaf und darauf gehen wir in Abschnitt 13.8 ein.

Als *Schnittstellen über den Webbrowser* stehen Ihnen als Variante eins die Paketsuche über die Debian-Webseite zur Verfügung [Debian-Debtags-Search]. Bei den Suchergebnissen werden die Debtags des jeweiligen Pakets von vornherein mit angezeigt (siehe Abbildung 13.2). Variante zwei ist der *Debtags Editor*, welchen wir in Abschnitt 13.7 genauer besprechen.

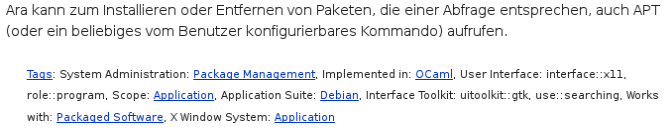

Abbildung 13.2: Debian Tags zum Paket *xara-gtk*

13.5 Vergebene Schlagworte anzeigen

13.5.1 Auf der Kommandozeile

Hier ist das Programm `debtags` mit Hilfe der Unterkommandos `cat`, `show` und `tag` am besten geeignet. Während `cat` für *alle* Pakete deren hinterlegte Schlagworte auflistet, erfordern `show` und `tag` als weitere Angabe im Aufruf noch den Namen des gewünschten Pakets. Zu diesem stellt `debtags` dann alle Informationen detailliert dar. In den ersten beiden Fällen kommt zusätzlich das UNIX-Werkzeug `grep` ins Spiel, welches Ihnen aus der Ausgabe jeweils die spezifische Zeile mit den Debtags herausfischt. Im ersten Fall benötigt `grep` den Paketnamen, hier beispielhaft am Paket *xpdf* zu sehen.

Auflistung der vergebenen Schlagwörter samt Wert für das Paket xpdf anhand von debtags cat

```
$ debtags cat | grep xpdf
xpdf: implemented-in::c++, interface::x11, role::program, scope::application, uitoolkit:: ↩
    motif, use::viewing, works-with-format::pdf, works-with::text, x11::application
$
```

Im zweiten Fall ist lediglich die Zeile mit dem Stichwort `Tag` interessant – nachfolgend wiederum beispielhaft am Paket *xpdf* zu sehen.

Auflistung der vergebenen Schlagwörter samt Wert für das Paket xpdf anhand von debtags show

```
$ debtags show xpdf | grep Tag
Tag: implemented-in::c++, interface::x11, role::program, scope::application, uitoolkit:: ↩
    motif, use::viewing, works-with-format::pdf, works-with::text, x11::application
$
```

Im dritten Fall erhalten Sie eine Auflistung mit einem Schlagwort pro Zeile in alphabetisch aufsteigender Abfolge, was bspw. im Rahmen einer Weiterverarbeitung durch Skripte nützlich ist. Für das Paket `xpdf` sieht das wie folgt aus:

Auflistung der vergebenen Schlagwörter samt Wert für das Paket xpdf anhand von debtags tag ls

```
$ debtags tag ls xpdf
implemented-in::c++
interface::x11
role::program
scope::application
uitoolkit::motif
use::viewing
works-with-format::pdf
works-with::text
x11::application
$
```

Obige Ausgaben besagen, dass *xpdf* als X11-Programm mit einer entsprechenden graphischen Schnittstelle einsortiert ist (`interface::x11`, `x11::application`), welches zu den Anwendungen zählt (`role::program`, `scope::application`) und genauer gesagt zu den Betrachtern für PDF und Text gehört (`use::viewing`, `works-with-format::pdf`, `works-with::text`). Ersichtlich ist außerdem, dass *xpdf* das Motif-Toolkit verwendet (`uitoolkit::motif`) und in der Sprache C++ entwickelt wurde (`implemented-in::c++`).

Benötigen Sie zu einem Paket stattdessen lediglich die Namen der vergebenen Schlagworte ohne deren konkreten Wert, erreichen Sie das über den Aufruf von `debtags` mit Hilfe dessen Unterkommandos `grep` und des Schalters `--facets`, welches Sie wiederum über das UNIX-Kommando `grep` und dem Paketnamen filtern. Nachfolgend sehen Sie das für die Recherche zum Paket *apt-move* [Debian-Paket-apt-move].

Auflistung der vergebenen Schlagwörter ohne Wert für das Paket apt-move

```
$ debtags grep --facets | grep apt-move
apt-move: hardware, implemented-in, interface, role, scope, suite, use, works-with
$
```

13.5.2 Integration in `aptitude`

Sofern das Paket *debtags* auf Ihrem System installiert ist, stellt auch `aptitude` die hinterlegten Schlagworte als Zusatzinformationen zum gerade von Ihnen ausgewählten Paket dar. Abbildung 13.3 zeigt dies ebenfalls für den PDF-Betrachter `xpdf` und das gleichnamige Paket dazu.

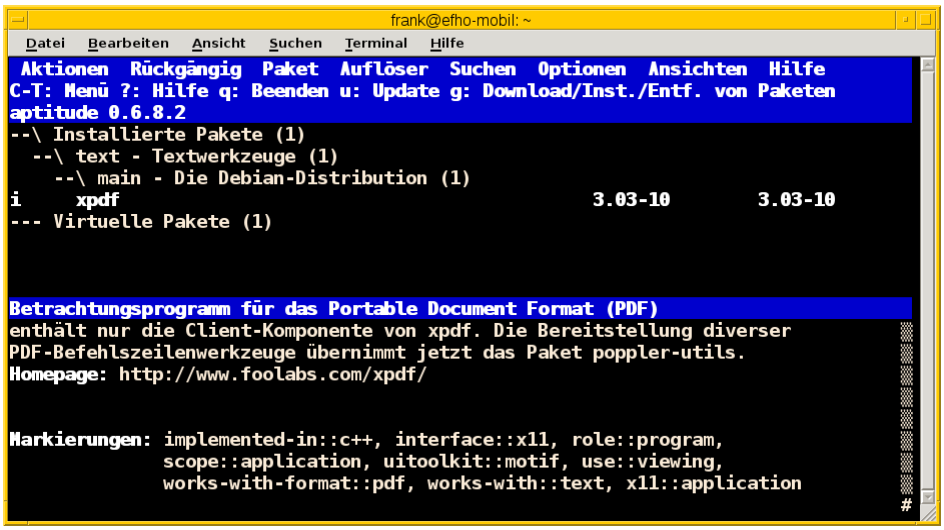

Abbildung 13.3: Darstellung der Schlagworte zum Paket *xpdf* in `aptitude`

Dass die Übersetzung von Begriffen keine triviale Aufgabe ist, sehen Sie in der Paketbeschreibung. Anstatt von Schlagwörtern wird in der deutschen Version von `aptitude` der Begriff Markierungen benutzt, was ebenfalls zutreffend ist.

13.5.3 Graphische Programme

Debtags zeigen Ihnen PackageSearch und Xara an, ebenso die bereits oben genannten Programme aus dem Paket *goplay*. Alle Werkzeuge haben gemeinsam, dass die Debtags-Informationen auf der rechten Seite der Benutzeroberfläche zu finden sind – entweder in der oberen oder unteren Hälfte.

Bei PackageSearch tragen Sie zuerst das gewünschte Paket über das Suchfeld unten rechts ein und sehen danach die Debtags-Informationen im darunter angeordneten Reiter `Details`. Bei Xara ist Ihr Ziel das Informationsfeld oben rechts, welches gefüllt wird, nachdem Sie zuvor links unten das Paket und danach links oben den Eintrag `Tag` aus der Liste der Paketdetails ausgewählt haben.

Abbildung 13.4 zeigt Ihnen das Suchergebnis zum Paket *ara*. Dieses Softwarepaket wurde in den beiden Sprachen C++ und Ocaml entwickelt (`implemented-in::c++`, `implemented-in::ocaml`), verfügt über eine Kommandozeilenschnittstelle (`interface::commandline`) und zählt ebenfalls zu den Anwendungen (`role::program`, `scope::application`). `ara` dient zur Suche in den Softwarepaketen (`use::searching`, `works-with::software-package`) und ist mit dem Debtag `suite::debian` als Debian-spezifisches Programm gekennzeichnet.

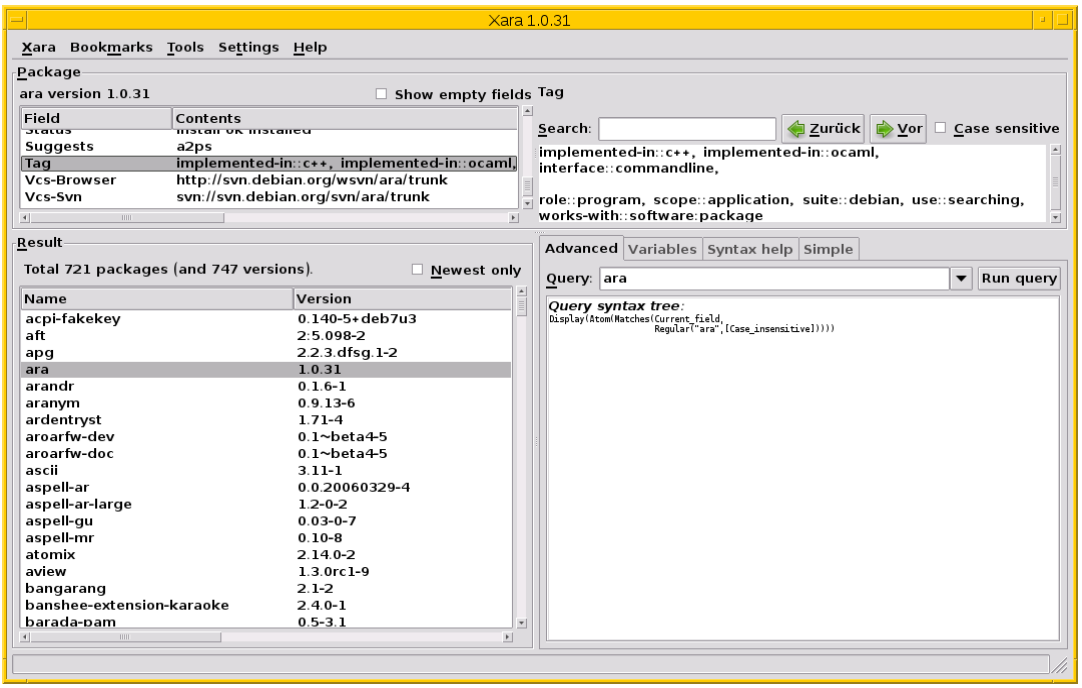

Abbildung 13.4: Darstellung der Schlagworte zum Paket *xara-gtk* in Xara

13.5.4 Über den Webbrowser

Für diese Aktivität ist zunächst die Webseite des Debian-Projekts und insbesondere der Unterpunkt `Paketsuche` [Debian-Paketsuche] ein zentraler Anlaufpunkt. Wie bereits eingangs erwähnt, listet Ihnen die Paketsuche automatisch die vergebenen Schlagwörter für ein Paket auf (siehe dazu Abbildung 13.2).

Ebenso aufschlussreich und noch deutlich ausführlicher ist der *Debtags Editor*. Darüber sehen Sie nicht nur die vergebenen Schlagwörter für ein Paket, sondern korrigieren diese bei Bedarf direkt. Mehr dazu erfahren Sie in Abschnitt 13.7.

13.6 Suche anhand der Schlagworte

13.6.1 Über die Kommandozeile

Hier spielt wiederum das bereits zuvor eingesetzte Werkzeug `debtags` seine Stärken aus – diesmal mit den beiden Unterkommandos `grep` und `search` gefolgt von Schaltern und einer Liste der Suchbegriffe. Während der Aufruf von `debtags grep` dabei lediglich die gesamte Zeile aus der Paketdatenbank extrahiert, in der der von Ihnen gewählte Suchbegriff vorkommt, liefert Ihnen `debtags search` lediglich den Paketnamen und die Kurzbeschreibung zum Paket in einer einzigen Zeile zurück. Je nach konkretem Anwendungsfall ist das ausgesprochen praktisch.

Dabei bestehen die **Suchbegriffe** aus einem Debtags-Eintrag. Dieser Eintrag besteht wiederum aus drei Teilen – einem Schlagwort (Tag), gefolgt von zwei Doppelpunkten (`::`) als Trennzeichen und dem gewünschten Wert für das vorher benannte Tag. Korrekt sind bspw. `role::program`, `suite::debian` und `use::searching`. Im ersten Beispielaufruf sehen Sie eine Suche nach den Paketen, die mit dem Datenformat PDF umgehen können und daher entsprechend mit dem Schlagwort `works-with-format::pdf` markiert sind. Da die Liste recht lang ist, umfasst das nachfolgende gezeigte Ergebnis lediglich die beiden Pakete *pdfgrep* und *pdfjam* mit ihren jeweiligen Schlagworten.

```
$ debtags grep "works-with-format::pdf"
...
pdfgrep: implemented-in::c++, role::program, scope::utility, use::searching, works-with- ↩
    format::pdf, works-with::file
```

```
pdfjam: implemented-in::shell, interface::commandline, role::program, scope::utility, use:: ↩
    converting, works-with-format::pdf, works-with::text
...
$
```

Wie bereits oben angesprochen, sind im Aufruf ebenfalls verschiedene Schalter zulässig. Geht es Ihnen ausschließlich um die Paketnamen, ist für Sie der Schalter `--names` interessant. Damit beschränken Sie die Ausgabe nur auf die Liste der Paketnamen. Die vergebenen Schlagworte werden nicht mit ausgegeben. Die untenstehende Ausgabe enthält eine Auswahl der Pakete, die Debian-spezifisch sind und daher das Schlagwort `suite::debian` tragen.

```
$ debtags grep --names "suite::debian"
adduser
alien
approx
apt
apt-build
apt-cacher
apt-cacher-ng
apt-doc
apt-dpkg-ref
apt-file
...
$
```

Über den Schalter `-i` (Langform `--invert`) erhalten Sie das umgekehrte Suchergebnis, d.h. alle Treffer, in denen ihr Suchbegriff *nicht* enthalten ist. Benötigen Sie zu einem Paket stattdessen lediglich die Namen der vergebenen Schlagworte ohne deren konkreten Wert, erreichen Sie das über den Schalter `--facets` (siehe dazu Abschnitt 13.5).

Für die Suche anhand mehrerer Schlagworte kombinieren Sie diese im Aufruf mit zwei Kaufmanns-Und. Im nachfolgenden Beispiel sehen Sie eine Suche nach den Spielen, die einerseits X11-tauglich sind und andererseits als Simulation einsortiert wurden. Daher umfasst die Recherche die beiden Tags `interface::x11` und `game::simulation`.

```
$ debtags search "game::simulation && interface::x11"
billard-gl - 3D billiards game
cultivation - game about the interactions within a gardening community
foobillard - 3D billiards game using OpenGL
gtkpool - simple pool billiard game written with GTK+
libopenscenegraph-dev - 3D scene graph, development files
libopenscenegraph80 - 3D scene graph, shared libs
lincity-ng - City simulator game with polished graphics
oolite - space sim game, inspired by Elite
opencity - 3D city simulator game
openssn - modern submarine tactical simulator
openttd - reimplementation of Transport Tycoon Deluxe with enhancements
pinball-dev - Development files for the Emilia Pinball Emulator
searchandrescue - fly aircraft to search (for) and rescue people in distress
simutrans - transportation simulator
singularity - game where one becomes the singularity
stormbaancoureur - simulated obstacle course for automobiles
$
```

An dieser Stelle hilft Ihnen auch das Paket *dctrl-tools* [Debian-Paket-dctrl-tools] weiter – jetzt jedoch mit dem Programm `grep-debtags`. Mit dem nachfolgendem Aufruf erhalten Sie eine Liste aller verfügbaren Pakete zu leichtgewichtigen Webbrowsern, die keinen Bezug zum Kool Desktop Environment (KDE) haben. Über die beiden Schalter `-sPackage` und `-d` reduzieren Sie die Ausgabe auf den Paketnamen und die einzeilige Paketbeschreibung, `-n` unterdrückt zusätzlich die Feldnamen. Mehrere Suchkriterien kombinieren Sie mittels `-a` für ein boolesches AND sowie `-a -!` für ein boolesches NAND.

```
$ grep-debtags -sPackage -d -n web::browser -a interface::x11 -a -! suite::kde
arora
simple cross platform web browser
```

```
chimera2
Web browser for X

dillo
Small and fast web browser

midori
fast, lightweight graphical web browser

xxxterm
Minimalist's web browser

$
```

13.6.2 Textoberfläche von `aptitude`

`aptitude` gruppiert die Pakete ebenfalls anhand ihrer Schlagworte. Diese zugegebenermaßen etwas versteckte Darstellung finden Sie im Programm unter Ansichten → Neuer Debtags-Browser (siehe Abbildung 13.5). Danach erhalten Sie eine Auswahlliste anhand der Debtags und wählen darüber ihre Pakete wie gewohnt aus.

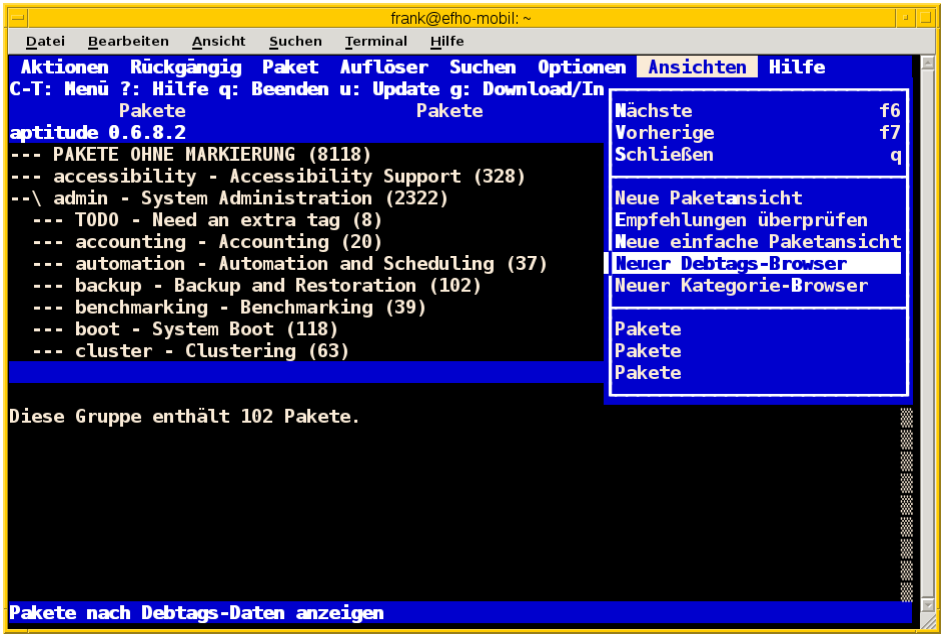

Abbildung 13.5: Auswahl des Debtags-Browsers in `aptitude`

13.6.3 Graphische Programme

In dieser Kategorie bleiben aus der Liste der Werkzeuge zur Paketverwaltung nur die beiden Kandidaten PackageSearch und Xara übrig (siehe Abbildung 13.6 und Abbildung 13.7). Bei ersterem stöbern Sie über Liste oben rechts und selektieren daraus die gewünschten Einträge. Bei Xara tragen Sie eine Suchanfrage in das Eingabefeld unter dem Reiter `Advanced` rechts unten ein und lösen danach mit `Run query` die Recherche aus. Alternativ wählen Sie den Reiter `Simple` aus und befüllen die (vielen) Eingabefelder entsprechend.

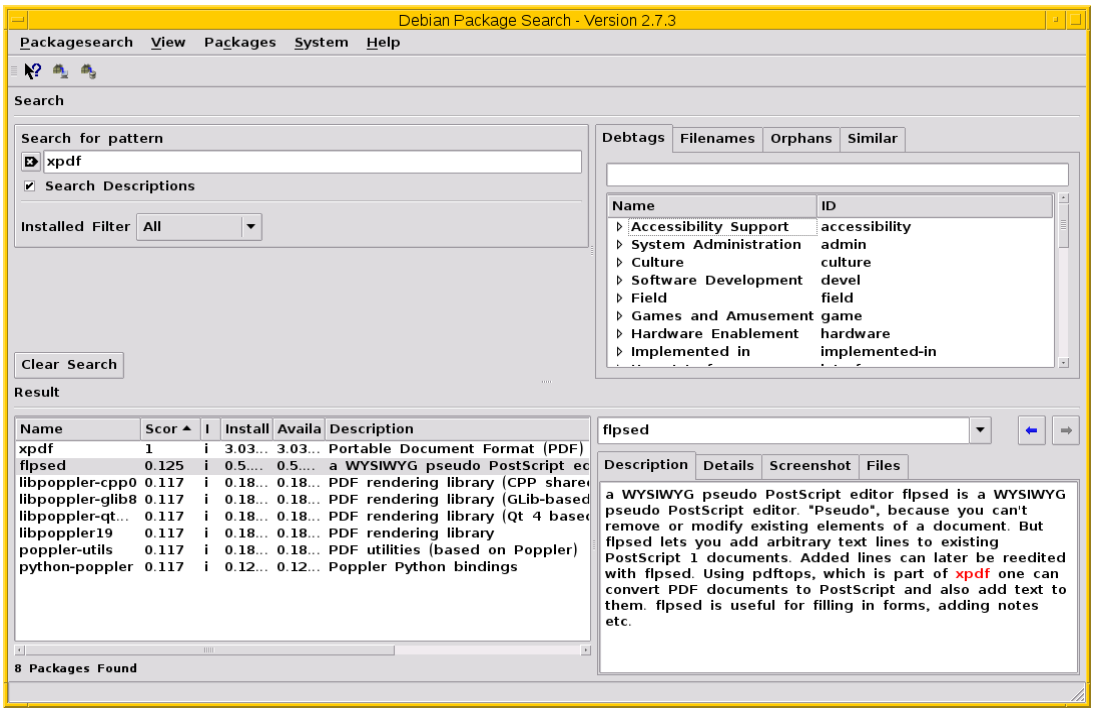

Abbildung 13.6: PackageSearch im Einsatz

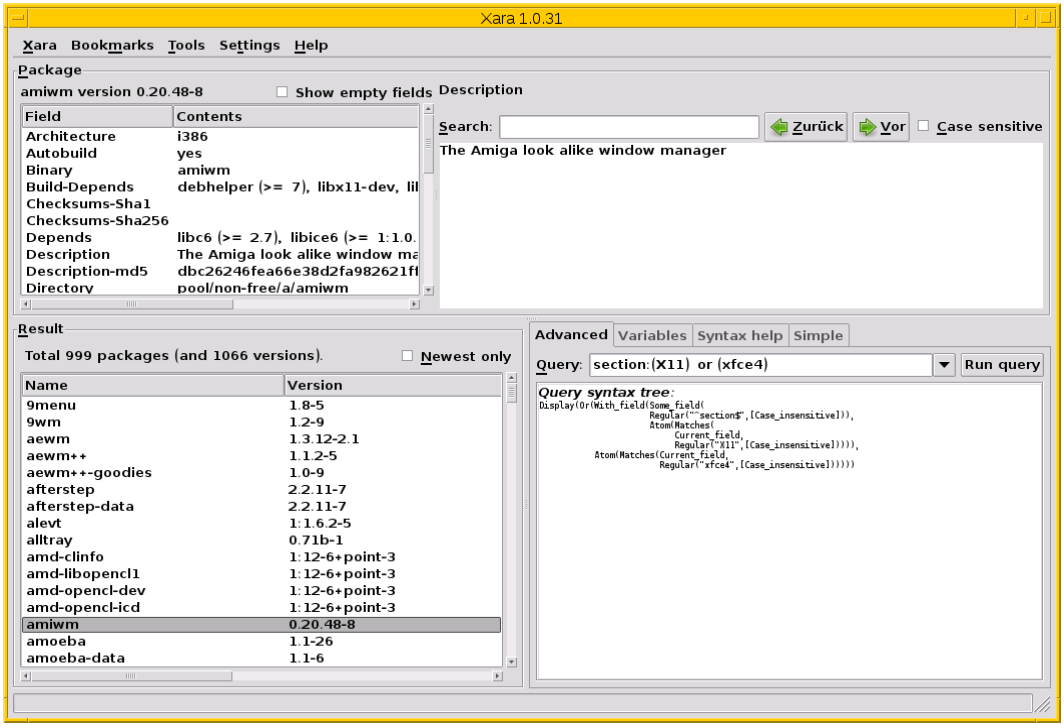

Abbildung 13.7: Xara im Einsatz

13.6.4 Suche über den Webbrowser

Eine webbasierte Recherche anhand der Debtags geht derzeit (noch) nicht über die Paketsuche, auch wenn die Debtags im Suchergebnis bereits angezeigt werden und anklickbar sind. Stattdessen stehen Ihnen der *Debtags Browser* [Debian-Debtags-Search] und die *Debtags Cloud* [Debian-Debtags-Search-By-Tags] zur Verfügung.

Die Schreibweise der Suchanfrage im Debtags Browser orientiert sich dabei an den Gepflogenheiten im Web. Das Formular nimmt eine direkte Eingabe der Debtags entgegen. In Abbildung 13.8 sehen Sie das Ergebnis einer Suche nach den Paketen, bei denen das Schlagwort `interface::commandline` hinterlegt ist und verifiziert wurde.

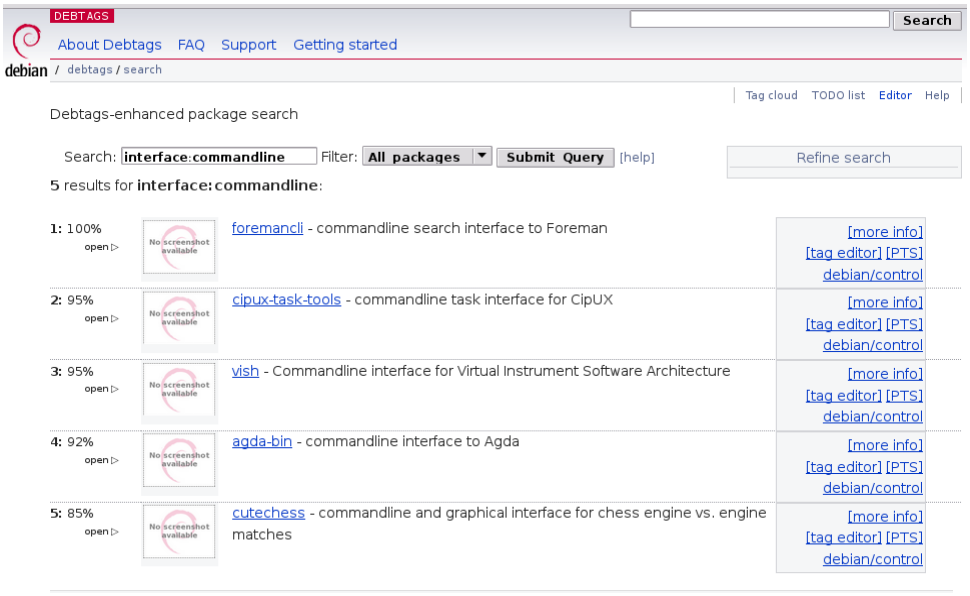

Abbildung 13.8: Suche anhand der Debtags über den Webbrowser

Die Recherche mit Hilfe der Debtags Cloud funktioniert etwas anders. Die Grundlage dafür bilden bereits überprüfte, validierte Schlagworte (sogenannte *reviewed tags*). Zunächst wählen Sie aus der „Wolke“ das gewünschte Schlagwort aus, woraufhin in der Ergebnisliste darunter alle Pakete aufgeführt werden, die mit diesem Schlagwort versehen sind (siehe Abbildung 13.9). Jeder Listeneintrag umfasst den Paketnamen, eine kurze Paketbeschreibung und alle bereits vergebenen Schlagwörter. Der Paketname des Listeneintrags ist dabei ein Link, der Sie direkt zum Debtags Editor bringt.

Aktivieren Sie einen Link in der „Wolke“ mit der Maus, erscheinen zwei zusätzliche Symbole – ein zustimmender und ein abwertender Daumen. Gleiches gilt für die Darstellung der ausgewählten Schlagworte in den beiden linken Spalten, die mit *Good Tags* bzw. *Bad Tags* betitelt sind. Über diese Symbole steuern Sie die Auswahl innerhalb der Wolke und grenzen ihren Suchbereich genauer ein. Ein zustimmender Daumen erweitert den Suchbereich, während ein abwertender Daumen den Suchbereich entsprechend verringert.

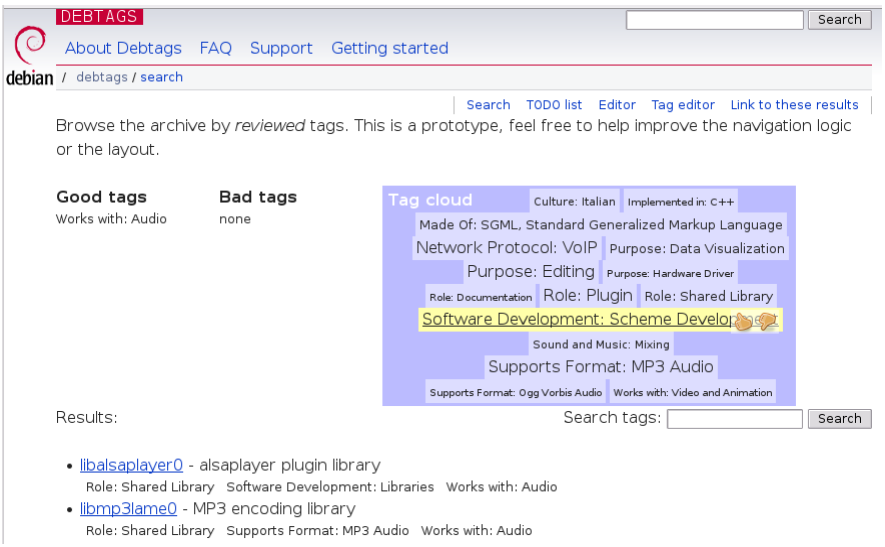

Abbildung 13.9: Auswahl der Pakete anhand der Debtags Cloud

13.7 Pakete um Schlagworte ergänzen

Nach dem derzeitigen Entwicklungsstand besteht keine Möglichkeit, über die Kommandozeile oder über graphische Werkzeuge die bereits vergebenen Schlagworte zu einem Paket zu verändern. Dafür ist der webbasierte *Debtags Editor* [Debian-Debtags-Editor] das Mittel der Wahl.

In Abbildung 13.10 sehen Sie diesen im Webbrowser Firefox/Iceweasel und darin beispielhaft die Informationen des Pakets *gimp* zur gleichnamigen Bildbearbeitungssoftware. Die Darstellung umfasst zwei Spalten – links die Informationen zum ausgewählten Paket und rechts die vergebenen Schlagwörter.

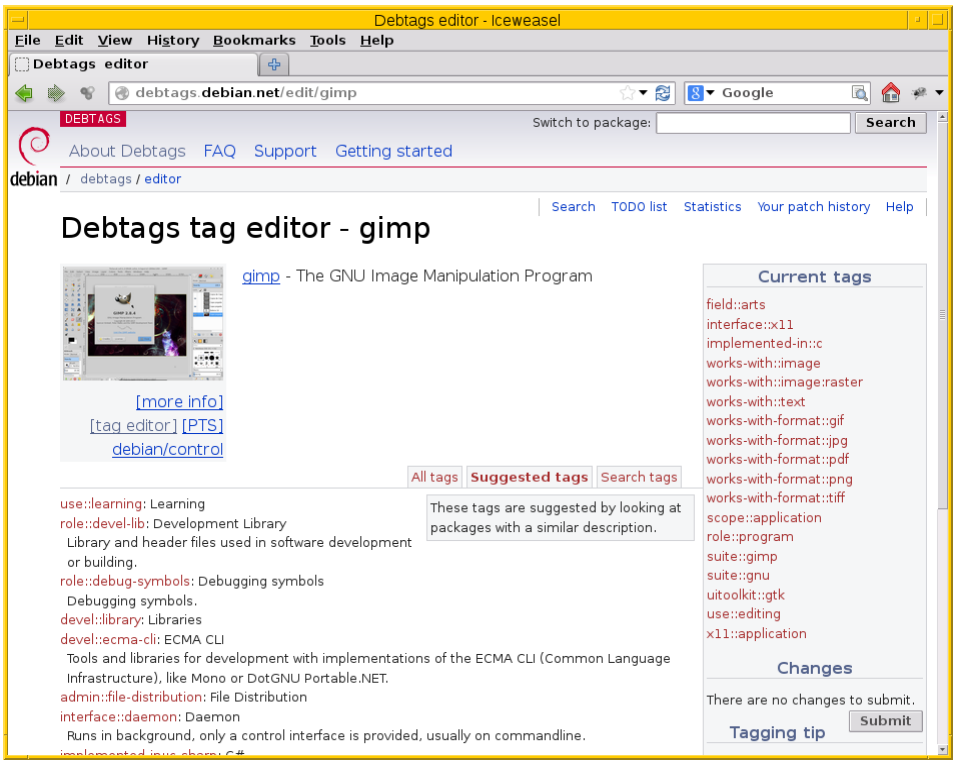

Abbildung 13.10: Webseite zum Debtags-Projekt mit den Informationen zum Paket *gimp*

Unter der Paketinformation stehen die verfügbaren und automatisch vorgeschlagenen Schlagwörter (Reiter `All tags` und `Suggested tags`). Letzteres umfasst die Schlagworte, die eventuell noch fehlen und die Sie ergänzen können, sofern Ihnen diese passend erscheinen. Die Vorschläge basieren auf einem Automatismus, der sich auf ähnliche Pakete und deren bereits bestehende Klassifikation stützt und lediglich Empfehlungen für noch fehlende Schlagwörter gibt. Daher ist es hilfreich, die Vorschläge kritisch zu prüfen und danach ggf. die gesamte Liste der Eigenschaften des Pakets durchzugehen. Dabei prüfen Sie am besten Schritt für Schritt, ob alle Schlagworte stimmig vergeben wurden. Alle verfügbaren Schlagworte verbergen sich hinter dem Reiter `All tags`.

In der rechten Leiste erscheinen zunächst die derzeit vergebenen Tags (bezeichnet mit `Current tags`). Darunter finden Sie die vorbereiteten Änderungen, bezeichnet mit `Changes`.

Nachdem Sie in der linken Spalte ein Schlagwort mit einem Mausklick ausgewählt haben, wird dieses zunächst nur zur Liste der vorbereiteten Änderungen hinzugefügt. Um eine dieser vorbereiteten Änderungen wieder rückgängig zu machen, wählen Sie das entsprechende Schlagwort aus der Liste der vorbereiteten Änderungen aus und entfernen es mit einem Klick darauf. Sind Ihre vorbereiteten Änderungen vollständig, klicken Sie auf den Knopf `Submit` und übertragen damit die Ergänzungen zum jeweiligen Paket zur Debtags-Datenbank. Danach sind Ihre Änderungen sofort für alle Benutzer verfügbar.

13.8 Verwendetes Vokabular bearbeiten und erweitern

Bislang lagen lediglich die Pakete und deren zugeordnete Schlagworte im Blickfeld. Nun rückt das dabei genutzte Vokabular in den Mittelpunkt, d.h. der dafür verwendete Wortschatz zur Klassifizierung der Pakete.

Die Federführung bei der Pflege übernimmt derzeit Enrico Zini, der Autor des Debtags-Projekts. Zu beobachten ist jedoch über die letzten Jahre, dass das Thema Verschlagwortung von den Benutzern verstärkt wahrgenommen wird und ein größeres Interesse besteht, daran mitzuwirken. Die vorgenommenen Veränderungen im Wortschatz reflektieren dabei einerseits den Wandel im Paketbestand und andererseits das gestiegene Bedürfnis der Debiananwender nach einer möglichst zielgenauen Recherchemöglichkeit zu den Paketen mit Hilfe passender sprachlicher Mittel.

13.8.1 Alle verfügbaren Schlagworte anzeigen

Auf der **Kommandozeile** erhalten Sie diese Informationen über den Aufruf `debtags tagcat`. Zu jedem Eintrag sehen Sie eine kürzere und eine ausführlichere Beschreibung, welche den Einsatzzweck des Schlagworts näher beleuchtet.

Auflistung der verfügbaren Schlagworte (Ausschnitt)

```
$ debtags tagcat
Tag: accessibility::input
Description: Input Systems
 Input Systems
Applies to input methods for non-latin languages as well as special input
systems.

...

$
```

13.8.2 Informationen zu Schlagworten anzeigen

Auf der **Kommandozeile** stehen Ihnen mehrere Möglichkeiten offen, weitere Informationen zu den Schlagworten zu erhalten. Das Werkzeug `debtags` zeigt Ihnen die Schlagworte an, die sich mit einem bestimmten Thema befassen. Dazu kennt es das Unterkommando `tagsearch`, welches Sie um einen weiteren Begriff ergänzen. Die nachfolgende Ausgabe zeigt Ihnen die hinterlegten Unterkategorien zum Schlagwort `mail`.

Anzeige aller verfügbaren Unterkategorien zum Schlagwort mail

```
$ debtags tagsearch mail
mail (facet) - Electronic Mail
mail::TODO - Need an extra tag
mail::delivery-agent - Mail Delivery Agent
mail::filters - Filters
mail::imap - IMAP Protocol
mail::list - Mailing Lists
mail::notification - Notification
mail::pop - POP3 Protocol
mail::smtp - SMTP Protocol
mail::transport-agent - Mail Transport Agent
mail::user-agent - Mail User Agent
protocol::fidonet - FidoNet
protocol::finger - Finger
protocol::imap - IMAP
protocol::nntp - NNTP
protocol::pop3 - POP3
protocol::smtp - SMTP
system::server - Server
works-with::mail - Email
$
```

Mit dem Unterkommando `tagshow` erhalten Sie weitere Informationen zu einem Schlagwort. Für die Facette `protocol::pop3` sieht das wie folgt aus:

Anzeige der Informationen zu einer spezifischen Facette, hier protocol::pop3

```
$ debtags tagshow "protocol::pop3"
Tag: protocol::pop3
Description: POP3
 POP3
Post Office Protocol, a protocol to download emails from a mail server,
designed for users that have only intermittent connection to the Internet.
```

```
In contrast to IMAP server, messages that are downloaded via POP3 are not
supposed to stay on the server afterwards, since POP3 does not support
multiple mailboxes for one account on the server.

Link: http://en.wikipedia.org/wiki/Post_Office_Protocol
Link: http://www.ietf.org/rfc/rfc1939.adoc
$
```

An **graphischen Werkzeugen** existiert nur das Programm `debtags-edit` [Debian-Paket-debtags-edit] von Enrico Zini. Aus Zeitmangel wurde das Werkzeug bislang nicht mehr von ihm weiterentwickelt und ist daher nicht mehr für die Veröffentlichung Debian *stable* verfügbar. Da die Quellen des Pakets aber frei zugänglich sind, dürfen Sie sich gern der Aufgabe annehmen, dieses Projekt fortzuführen.

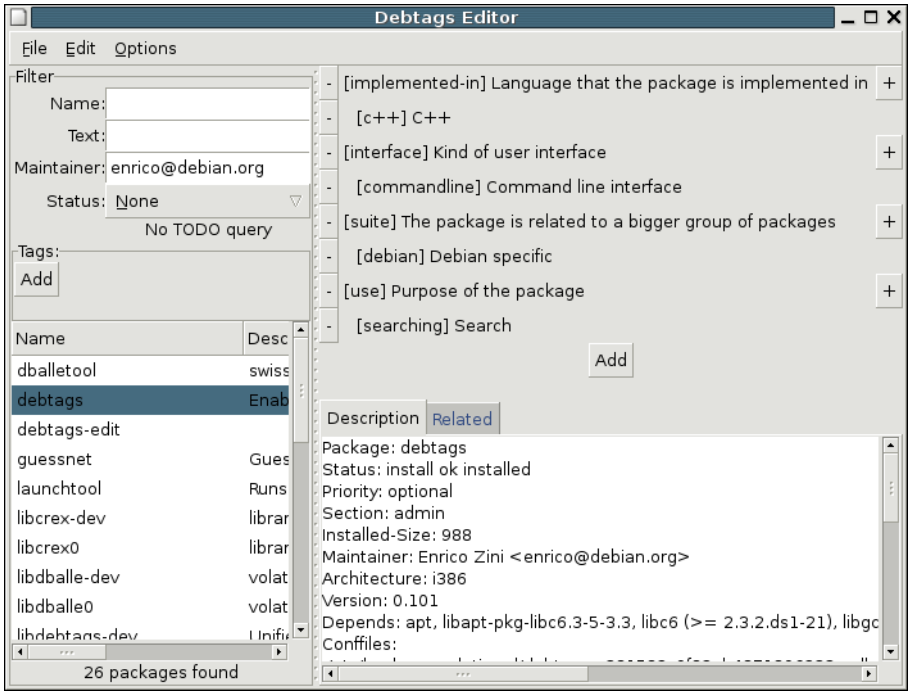

Abbildung 13.11: `debtags-edit` im Einsatz

Über die **Debtags-Webseite** sind Ihnen die Informationen zu einem Schlagwort ebenfalls zugänglich. Verlinkt werden dabei auch die Pakete, die mit diesem Schlagwort versehen sind (siehe Abbildung 13.12).

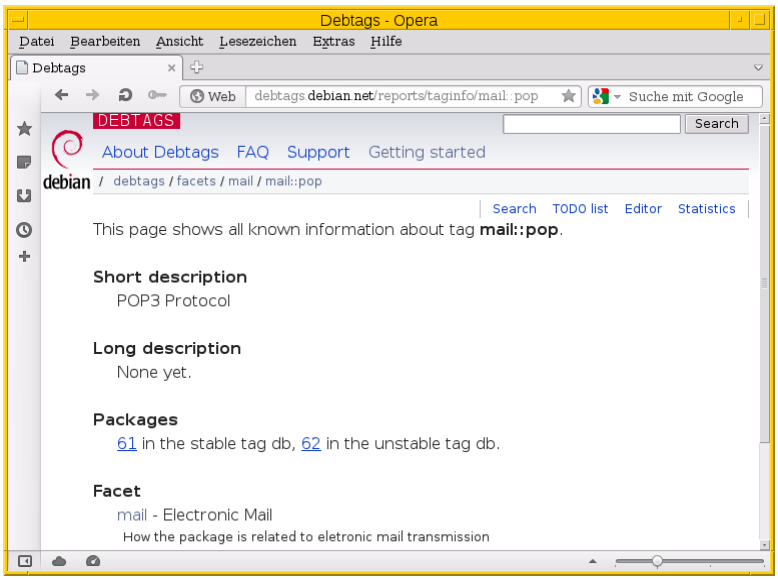

Abbildung 13.12: Informationen zur Facette `protocol::pop3`

13.8.3 Schlagworte bearbeiten

`debtags diff` *Dateiname* (Alternative `mkpatch`)
create a tag patch between the current tag database and the tag collection *Dateiname*. Standard input is used if filename is not specified.

`debtags submit` *Patchdatei*
upload the given patch file to the central tag repository. If *Patchdatei* is omitted, mail the local tag modifications (uses `debtags-submit-patch`).

`debtags tag add` *Paket Schlagworte*
Füge die angegebenen Schlagworte für das Paket in der Debtags-Datenbank hinzu.

`debtags tag rm` *Paket Schlagworte*
Entferne die angegebenen Schlagworte aus der Debtags-Datenbank für das genannte Paket.

`debtags update`
Collect package tag data from the sources listed in /etc/debtags/sources.list, then regenerate the debtags tag database and main index. It needs to be run as root.

`debtags-fetch`
fetch tag sources from /etc/debtags/sources.list

`debtags-submit-patch`
submit tag patches to http://debtags.debian.net

Kapitel 14

Mehrere Pakete in einem Schritt ändern

14.1 Mit `apt-get`

APT kennt mehrere Varianten für diesen Schritt. In Variante eins geben Sie beim Aufruf einfach mehrere Paketnamen in beliebiger Reihenfolge hintereinander an. Dabei erfolgt die Trennung der Paketnamen durch Leerzeichen. In Variante zwei schreiben Sie den Paketnamen nicht explizit aus, sondern als Muster. APT löst dieses Muster auf und liefert eine Liste der Paketnamen zurück, die auf das angegebene Muster passen.

In beiden Varianten erfolgt eine automatische Auflösung der Paketabhängigkeiten durch APT — es wird halt nur aufwendiger. Bei der Abarbeitung sortiert APT selbst und zwar automatisch so, dass die Paketabhängigkeiten möglichst problemlos aufgelöst werden können.

Installation der drei Pakete goplay, xara-gtk und debtags in einem Rutsch

```
# apt-get install goplay xara-gtk debtags
...
#
```

14.2 `aptitude`

Im Vergleich zu APT hat `aptitude` deutlich mehr auf dem Kasten. Es versteht bspw. verschiedene, zusätzliche **Optionen beim Aufruf**. Diese Optionen sind identisch zu den Tasten, die Sie in der Textoberfläche bei den Vormerkungen anwenden (siehe Kapitel 11).

Tabelle 14.1: Zusätzliche Optionen für den Aufruf bei `aptitude`

Taste	Bedeutung
`+paket`	Paket installieren (*install*)
`-paket`	Paket entfernen (*remove*)
`_paket`	Paket vollständig inklusive Konfigurationsdateien entfernen (*purge*)
`:paket`	Paketversion behalten (*keep*)
`=paket`	Paketversion dauerhaft beibehalten (*hold*)

Jeder der nachfolgend genannten Aufrufe sorgt dafür, dass Sie das Paket *goplay* installieren, *xara-gtk* deinstallieren und *debtags* so belassen (es auf *hold* setzen). Bitte beachten Sie, dass diese Zusatzoptionen stets in Bezug zu den Unterkommandos zu sehen sind.

Installation mit aptitude und Zusatzoptionen

```
# aptitude install goplay -xara-gtk =debtags
# aptitude remove xara-gtk +goplay =debtags
```

- ToDo: Umfangreich: `aptitude` (6)
  - Tabelle welche gibts
  - Änderungen am [Y/n/?] Prompt

Kapitel 15

Ausgewählte Pakete aktualisieren

In Abschnitt 8.39 beleuchteten wir, wie Sie alle bereits bestehenden Pakete auf ihrem Linuxsystem aktualisieren. Das hilft Ihnen dabei, dass alle Pakete aus einem in etwa ähnlichen Veröffentlichungszeitraum stammen und die Abstimmung der Pakete aufeinander weitestgehend konfliktfrei bleibt.

Nun zeigen wir Ihnen, wie Sie nur einzelne, ausgewählte Pakete auf einen bestimmten Stand bringen. Dieses Vorgehen bringt es mit sich, dass die Anzahl der Softwarepakete zunimmt, die explizit auf einem ausgewählten Stand gehalten werden bzw. werden müssen, um die Paketabhängigkeiten zu erfüllen. Für die Paketverwaltung heißt das, dass bei einer Veränderung des Paketbestands mehr und insbesondere komplexere Bedingungen zu berücksichtigen sind. Sie als Betreuer planen daher sicherheitshalber etwas mehr Zeit für die Wartung ein.

15.1 Nur ein einzelnes Paket aktualisieren

Im Alltag sind häufig für mehrere Pakete gleichzeitig Aktualisierungen verfügbar. Sowohl APT, als auch `aptitude` gestatten es Ihnen daher, nur die Pakete zu erneuern, die Sie wünschen. Stets werden dabei die Paketabhängigkeiten berücksichtigt und nur die Softwarepakete mit einbezogen, die es betrifft.

Auf der **Kommandozeile** verstehen `apt-get` und `aptitude` die beiden Unterkommandos `upgrade` und `dist-upgrade`, jeweils gefolgt von einer Liste von Paketnamen. Ältere Versionen von APT bis Version XYZ können noch nicht damit umgehen und ignorieren diese Liste.

Beispiel für Einzelaktualisierung mit apt-get upgrade

```
ToDo: Beispiel
```

Automatische Aktualisierung bei der Installation
Ist ein Paket bereits installiert und Sie führen erneut das Kommando `aptitude install Paketname` aus, wird es nach Möglichkeit durch eine neuere Version ersetzt. Es entspricht in diesem Fall dem Aufruf `aptitude upgrade Paketname`.

`aptitude` mit seinen Unterkommandos `safe-upgrade` und `full-upgrade` nimmt hingegen schon länger Parameter entgegen. Dabei sind nicht nur Paketnamen, sondern auch Suchausdrücke möglich. Bspw. erneuert der Aufruf `aptitude full-upgrade '?section(libs)'` alle aktualisierbaren Pakete aus der Kategorie *libs* (Bibliotheken).

Beispiel für Einzelaktualisierung mit aptitude safe-upgrade

```
ToDo: Beispiel
```

- Durchführung über die **Textoberfläche** von `aptitude`

  - Zweig/Kategorie Sicherheitsaktualisierungen
    * Paket markieren mit -
    * Paketvorschau mit **g**
    * Paketaktualisierung mit **g**
  - ToDo: da muss aber noch mehr sein ....
- Durchführung bei **Synaptic** (siehe Abschnitt 6.4.1)
  - Auswahl der Kategorie Aktualisierbar (Upstream)
  - Rechtsklick auf Eintrag in der Paketliste
  - Auswahl Zum Aktualisieren vormerken

15.2 Aktualisierung mit Wechsel der Veröffentlichung

Frage/Problem: Ich möchte alle Abhängigkeiten von *kdegames* von *unstable* auf *experimental* heben (da es noch kein neues kdegames-Metapaket gibt, das von diesen Versionen abhängt und daher `apt-get -t experimental kdegames` gar nix macht).

Bisherige Lösung:

Aufruf zur Auswahl

```
apt-get install -t experimental $( apt-cache depends kdegames | awk '{ print$4 }' )
```

Status:

- geht, ist aber umständlich
- kann man das einfacher machen? (ist eine Art pinning für ein einzelnes Paket)

Kapitel 16

Ausgewählte Pakete nicht aktualisieren

Die Paketverwaltung muss darüber Bescheid wissen, wenn Sie ein ausgewähltes Paket auf dem aktuellen Stand belassen möchten. Dazu existiert das Paketflag *hold*, welches Sie in dem Fall explizit festklopfen müssen (siehe auch Paketflags unter Abschnitt 2.15). Haben Sie das Flag gesetzt, wird das Paket somit nicht weiter aktualisiert oder gelöscht, sondern in dem derzeit bestehenden Zustand und der Version gehalten.

Ergebnis dieses Vorgehens ist, dass das Paket auf Ihrem Linuxsystem so erhalten bleibt, wie es aktuell ist. Es gibt keine Veränderungen bzgl. genutzter Schnittstellen und Abhängigkeiten zu anderen Paketen. Das ist insbesondere dann sinnvoll, wenn eine ganz bestimmte Komponente benötigt wird, die sich nicht verändern darf. Hintergrund können bspw. Softwareunverträglichkeiten, Schnittstellenänderungen oder auch noch nicht abgeschlossene Anpassungen sein. Sind diese umgesetzt, kann das Paketflag dann wieder entfernt werden.

Dieses Vorgehen hat Fallstricke. Je länger Sie warten und das Paket zurückhalten, umso stärker entwickelt sich das Drumherum weiter. In Folge heißt das, dass Aktualisierungen schwieriger und insbesondere aufwendiger werden. Bedenken Sie diesen Schritt daher gut.

16.1 Auf der Kommandozeile

Um ein Paket auf einem bestimmten Versionsstand zu halten, nutzen Sie das Werkzeug `apt-mark`. Es kennt das Unterkommando `hold`, um damit das dazugehörige Paketflag zu setzen, und `unhold`, um diese Festlegung zu widerrufen. Beide Aufrufe akzeptieren als Parameter eine Liste der Paketnamen. Die nachfolgenden Ausgaben zeigen das Vorgehen für das Paket *wireshark*.

Aufruf von apt-mark zum Setzen der Markierung hold für das Paket wireshark

```
# apt-mark hold wireshark
wireshark auf Halten gesetzt.
#
```

Entfernen der Markierung hold für das Paket wireshark mittels apt-mark

```
# apt-mark unhold wireshark
Halten-Markierung für wireshark entfernt.
#
```

`apt-mark` kennt zudem das Unterkommando `showhold`. Damit lassen Sie sich auflisten, welche Pakete auf Ihrem System derzeit auf *hold* gesetzt sind. Geben Sie keine Pakete als Parameter an, werden alle Pakete untersucht. Ansonsten berücksichtigt `apt-mark` nur die von Ihnen spezifizierten Pakete.

Auflistung aller Pakete, die gehalten werden

```
# apt-mark showhold
wireshark
#
```

16.2 Textoberflächen

- `aptitude`

16.3 Graphische Programme

- Synaptic

Kapitel 17

Fehlende Pakete bei Bedarf hinzufügen

17.1 Neue Hardware

Die Auswahl der installierten Pakete auf Ihrem System orientiert sich an den Hardwarekomponenten, die im System verbaut wurden und in Benutzung sind. Ändert sich daran etwas — bspw. Hardware wird ausgetauscht oder hinzugefügt — werden auch andere Module zur Unterstützung dieser Komponente benötigt.

Etwas mühselig ist es, durch eigene Experimente Gewissheit zu bekommen, welche Pakete aufgrund des Komponentenwechsels entfernt und ergänzt werden müssen. In diesem Fall kommt das Paket *isenkram* [Debian-Paket-isenkram] des norwegischen Entwicklers Petter Reinholdtsen ins Spiel. Dieses hat derzeit noch den Status *testing* und steht unter sehr aktiver Entwicklung [Isenkram-Reinholdtsen]. Hinter Isenkram verbirgt sich eine Art Benachrichtigungsdienst, der überprüft, ob die benötigten Pakete für die neue Hardware bereits auf dem System installiert sind. Falls nicht, wird dieses (automatisch) nachgezogen. Dazu klinkt es sich als zusätzliches Modul in die Konfiguration von *tasksel* ein (siehe Abschnitt 6.3.1).

ToDo:

- wie funktioniert das
  - wie klinkt es sich in *tasksel* ein
- was macht das alles
- wie benutzt man das
  - Benutzerschnittstelle: Paket *isenkram-cli* [Debian-Paket-isenkram-cli]

17.2 Neue Software

- Paket *auto-apt* ([Debian-Paket-auto-apt])
- Kurzfassung: "package search by file and on-demand package installation tool auto-apt checks the file access of programs running within its environments, and if a program tries to access a file known to belong in an uninstalled package, auto-apt will install that package using apt-get. This feature requires apt and sudo to work. It also provides simple database to search which package contains a requesting file."
- Beschreibung aus dem Debian-Anwenderhandbuch:
  - "auto-apt dient zur Installation von Programmen „bei Bedarf“. Dies kann beispielsweise bei der Software-Entwicklung oder auch schon beim einfachen Übersetzen eines neuen Kernels sinnvoll sein. Mitunter fehlt auf frisch installierten Debian Systemen noch das Paket bin86, das für das Erzeugen eines Kernels benötigt wird. Um automatisch die fehlenden Pakete zu installieren, stellen Sie einfach dem Aufruf von make das Kommando auto-apt voran: auto-apt make bzImage. Werden nun während des Durchlaufs fehlende Programme festgestellt, so ermittelt auto-apt, zu welchem Paket diese gehören, und installiert die fehlenden Pakete inklusive aller Abhängigkeiten."

- Aufruf: `auto-apt run Kommando`
  - werden zusätzliche Dateien benötigt, fragt `auto-apt`, ob diese nachinstalliert werden sollen
- Basis: kleine Datenbank, die aktuell gehalten sein sollte
  - Kommandos dazu:
    * `auto-apt update`
    * `auto-apt updatedb`
    * `auto-apt update-local`

Kapitel 18

Alternative Standard-Programme mit Debians Alterr System

Moderne Benutzeroberflächen regeln die Auswahl des Standard-Webbrowsers und des Standard-Editors über XDG[1]. Die Basis dafür bildet die Zuordnung der Anwendungen über MIME (engl. *MIME Applications Associations*) [mime-applications-associations] und im Speziellen die Liste der Standardanwendungen (engl. *Default Applications*) [mime-applications-associations-default-applications]. Zur Konfiguration steht Ihnen meist eine passende grafische Oberfläche zur Verfügung.

Ähnliches leisten die Programme `select-editor`, `sensible-browser`, `sensible-editor` und `sensible-pager` aus dem Paket *sensible-utils* [Debian-Paket-sensible-utils]. Diese werten primär die Umgebungsvariablen wie `$EDITOR`, `$BROWSER` und `$PAGER` aus und leiten daraus die Programmauswahl ab. Viele Einzelanwendungen, die wiederum andere Standardprogramme aufrufen, ermitteln bspw. die Information nicht selbst, welchen Editor Sie als Benutzer bevorzugen und verwenden. Sie verlassen sich stattdessen auf die Rückgabewerte, die Ihnen an dieser Stelle vom Werkzeug `sensible-editor` geliefert werden. Gleiches gilt für die anderen drei Werkzeuge.

Ebenso funktioniert das auch in Ihrem Terminal. Rufen Sie darin die Programme namens `editor` oder `x-www-browser` auf, werden ebenfalls diese Variablen ausgewertet und das darüber referenzierte Werkzeug ausgeführt.

Leider funktionieren diese beiden o.g. Arten der Auswahl des Standardprogramms nicht für alle Einsatzzwecke, sei es, weil es keinen passenden MIME-Typ für eine bestimmte Aufgabe gibt — z.B. das Standard-Desktop-Hintergrundbild — oder weil niemand bislang eine allgemein gültige Umgebungsvariable dafür definiert hat. Letzteres betrifft bspw. das Ballerspiel Doom, für das mehrere, unterschiedliche Spieleengines verfügbar sind.

Aus diesem Grund ermöglicht Debian Ihnen als Systembetreuer zusätzlich zu den bereits vorher beschriebenen Varianten auch noch die Festlegung eines sinnvollen Standardprogramms über das Debian Alternativen-System. Basierend auf den lokal installierten Werkzeugen legen Sie dieses Standardprogramm manuell fest oder lassen dieses automatisch auswählen.

Das Alternativen-System ist eine Debian-Eigenheit, die für viele Programme und -Pakete existiert. In RedHat/Fedora wurde das System in leicht veränderter Form übernommen [Debian-Wiki-Alternatives]. Ubuntu präferiert hingegen XDG und greift vor allem dann auf das Debian Alternativen-System zurück, wenn es die Pakete unverändert von Debian übernimmt.

Es basiert auf symbolischen Verweisen (kurz *Symlink*) nach und in das Verzeichnis `/etc/alternatives/` sowie gemäß der vom Paketbetreuer zuvor festgelegten Priorität pro Alternative. Folgen Sie den Vorgaben, wird dabei stets das Programm ausgeführt, welchem die höchste Priorität zugeordnet ist. Eine manuelle Festlegung hebt die Vorgaben auf.

Das Werkzeug zur Verwaltung dieser Symlinks auf der Kommandozeile heißt `update-alternatives`, für die graphische Benutzerschnittstelle steht Ihnen *galternatives* aus dem gleichnamigen Paket bereit [Debian-Paket-galternatives]. In Abbildung 18.1 sehen Sie exemplarisch die Möglichkeiten für die Gruppe `x-www-browser`.

[1] X Desktop Group (XDG), seit dem Jahr 2000 freedesktop.org

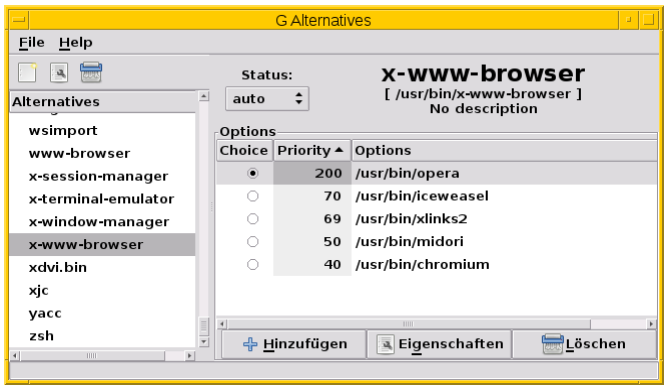

Abbildung 18.1: Auswahl mit Hilfe des Programms *galternatives*

18.1 Hintergrund: Warum alternative Standardprogramme?

In Debian stellen mehrere Programme bzw. Pakete eine ähnliche oder gleiche Fähigkeit bereit.

Dazu zählen auf der einen Seite sehr generische Funktionalitäten wie z.B. die eines Webbrowsers, eines (Text-)Editors oder eines Pagers, aber auch sehr spezifische Funktionalitäten, für die einfach mehrere Implementierungen vorliegen. Zu Letzeren gehören bspw. das Schweizer Taschenmesser für TCP/IP namens `netcat` oder die Skriptsprache `awk`. Letztere liegt in drei Varianten vor — in der eher umfangreicheren Implementation des GNU Projektes (`gawk`), in der kleineren und schnelleren Implementation von Mike Brennan (`mawk`) und in der originalen Awk-Implementation [awk].

Weiterhin gibt es von einigen Programmen mehr als eine Programmgeneration in Debian, so z.B. verschiedene Veröffentlichungen von GCC und Automake. In Debian 8 *Jessie* ist GCC in den beiden Versionen 4.8 und 4.9 enthalten, Automake in den Versionen 1.11 und 1.14. Einer der Gründe für die verschiedenen Versionen ist, dass manche Pakete in Debian z.B. nur mit einer bestimmten Generation eines Compilers kompiliert werden können. So werden z.B. alle Linuxkernel in Debian 8 *Jessie* mit GCC 4.8 kompiliert, während GCC 4.9 der Standardcompiler ist. Je nach Einsatzzweck eines Systems kann es entsprechend auch hilfreich sein, den Standardcompiler systemweit auf eine bestimmte Generation festzulegen.

Ein weiterer Grund für die Verwendung des Alternativen-Systems innerhalb desselben Binärpakets oder von Binärpaketen auf der Basis desselben Sourcepakets sind unterschiedliche Konfigurationen oder variierende Abhängigkeiten. Beispiel eins ist GNU Emacs, welcher in drei Varianten vorliegt:

- basierend auf dem Gimp Tool Kit (GTK+) und mit voller Unterstützung moderner Desktops (`emacs23` bzw. `emacs24`). Dieses Paket hängt u.a. von D-Bus ab.
- basierend auf dem schlankeren Lucid-Toolkit (`emacs23-lucid` bzw. `emacs24-lucid`). Mit Unterstützung für das X-Window-System, aber ohne allzuviele sonstige Abhängigkeiten.
- ganz ohne grafische Benutzeroberfläche (`emacs23-nox` bzw. `emacs24-nox`).

Ein zweites Beispiel ist der Windowmanager `dwm`, bei welchem die Konfiguration zum Kompilierzeitpunkt festgelegt wird. Das Paket *dwm* [Debian-Paket-dwm] enthält daher vier Programme mit einer jeweils unterschiedlichen Konfiguration — `dwm.default`, `dwm.maintainer`, `dwm.web` und `dwm.winkey`. Über das Alternativen-System legen Sie fest, welches davon verwendet wird, wenn Sie lediglich `dwm` aufrufen.

Viele Administratoren haben zudem sehr genaue Vorstellungen, welche Programme verwendet werden sollten, wenn sie unter dem generischen Programmnamen aufgerufen werden.

18.2 Standardprogramme anzeigen

Mit dem Aufruf `update-alternatives --get-selections` listen Sie alle generischen Programme oder Dateien auf, für die es Alternativen auf Ihrem lokalen System gibt. Ebenfalls mit ausgegeben werden dabei die aktuell ausgewählte Alternative

sowie die konkrete Auswahlform — automatisch anhand der installierten Pakete und Prioritäten oder manuell durch den lokalen Administrator.

Beispielausgabevon Axels Thinkpad und mit einer durchaus nicht ganz üblichen Auswahl von update-alternatives --get-selections (massiv gekürzt)

```
$ update-alternatives --get-selections
automake                        auto     /usr/bin/automake-1.14
awk                             auto     /usr/bin/gawk
c++                             auto     /usr/bin/g++
c89                             auto     /usr/bin/c89-gcc
c99                             auto     /usr/bin/c99-gcc
cc                              auto     /usr/bin/gcc
cpp                             auto     /usr/bin/cpp
csh                             auto     /bin/bsd-csh
de.multi                        manual   /usr/lib/aspell/de-alt.multi
desktop-background              auto     /usr/share/images/desktop-base/lines- ↩
    wallpaper_1920x1080.svg
desktop-background.xml          auto     /usr/share/images/desktop-base/lines.xml
desktop-grub                    auto     /usr/share/images/desktop-base/lines-grub.png
desktop-splash                  auto     /usr/share/images/desktop-base/spacefun-splash.svg
doom                            auto     /usr/games/chocolate-doom
dwm                             auto     /usr/bin/dwm.default
editor                          manual   /usr/bin/zile
emacs                           auto     /usr/bin/emacs24-x
emacsclient                     auto     /usr/bin/emacsclient.emacs24
ex                              auto     /usr/bin/nex
gnome-text-editor               auto     /usr/bin/leafpad
gnome-www-browser               auto     /usr/bin/opera
html2markdown                   auto     /usr/bin/html2markdown.py2
infobrowser                     auto     /usr/bin/info
jar                             auto     /usr/bin/fastjar
java                            auto     /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java
ksh                             auto     /bin/ksh93
locate                          auto     /usr/bin/mlocate
mp3-decoder                     auto     /usr/bin/mpg321
nc                              manual   /bin/nc.traditional
pager                           auto     /bin/less
rcp                             auto     /usr/bin/scp
rename                          auto     /usr/bin/file-rename
rlogin                          auto     /usr/bin/slogin
rsh                             auto     /usr/bin/ssh
rxvt                            manual   /usr/bin/urxvt
ssh-askpass                     manual   /usr/bin/ssh-askpass-fullscreen
telnet                          auto     /usr/bin/telnet-ssl
unison                          auto     /usr/bin/unison-latest-stable
unison-gtk                      auto     /usr/bin/unison-latest-stable-gtk
vi                              manual   /usr/bin/nvi
view                            manual   /usr/bin/nview
wesnoth                         auto     /usr/games/wesnoth-1.10
www-browser                     auto     /usr/bin/links2
x-cursor-theme                  manual   /etc/X11/cursors/crystalwhite.theme
x-session-manager               auto     /usr/bin/choosewm
x-terminal-emulator             manual   /usr/bin/uxterm
x-window-manager                manual   /usr/bin/ratpoison
x-www-browser                   manual   /usr/bin/conkeror
$
```

Welche Alternativen für ein generisches Kommando verfügbar sind, erfahren Sie mit dem Schalter `--list`. Nachfolgend sehen Sie das für die Skriptsprache Awk.

Ausgabe der verfügbaren Alternativen für die Skriptsprache Awk

```
$ update-alternatives --list awk
/usr/bin/gawk
/usr/bin/mawk
/usr/bin/original-awk
$
```

Über den Schalter `--display` erfahren Sie die derzeit festgelegte Alternative für ein generisches Kommando mitsamt den verfügbaren, weiteren Möglichkeiten und allen ebenfalls umgebogenen Referenzen auf dessen *Slaves*. *Slaves* sind weitere Dateien, die zu einem Programm dazugehören, bspw. die passenden Handbuchseiten (*Manual Pages*). Anhand des nachfolgenden Beispiels zu Awk verdeutlichen wir Ihnen das.

Ausgabe der ausgewählten und verfügbaren Alternativen für Awk

```
$ update-alternatives --display awk
awk - automatischer Modus
  Link verweist zur Zeit auf /usr/bin/gawk
/usr/bin/gawk - Priorität 10
  Slave awk.1.gz: /usr/share/man/man1/gawk.1.gz
  Slave nawk: /usr/bin/gawk
  Slave nawk.1.gz: /usr/share/man/man1/gawk.1.gz
/usr/bin/mawk - Priorität 5
  Slave awk.1.gz: /usr/share/man/man1/mawk.1.gz
  Slave nawk: /usr/bin/mawk
  Slave nawk.1.gz: /usr/share/man/man1/mawk.1.gz
/usr/bin/original-awk - Priorität 0
  Slave awk.1.gz: /usr/share/man/man1/original-awk.1.gz
Gegenwärtig »beste« Version ist »/usr/bin/gawk«.
$
```

Alternative Darstellung
Benötigen Sie stattdessen eine maschinenlesbare Ausgabe, hilft Ihnen in diesem Fall der Schalter `--query` weiter. Dabei werden die Blöcke in einer an den RFC 822 angelehnten Weise formatiert und zwischen den einzelnen Blöcken zusätzliche Leerzeilen eingefügt.

18.3 Standardprogramm ändern

Ist nur ein Paket installiert, welches für ein generisches Programm eine einzige Alternative anbietet, so wird automatisch dieses verwendet und es gibt keine Auswahl zur Konfiguration.

Hinweis, falls für ein generisches Programm nur eine Alternative installiert ist.

```
# update-alternatives --config emacs
Es gibt nur eine Alternative in Link-Gruppe emacs (die /usr/bin/emacs
bereitstellt): /usr/bin/emacs24-x
Nichts zu konfigurieren.
#
```

Installieren Sie hingegen mehrere Pakete, die alle eine Alternative für ein bestimmtes generisches Programm anbieten, so wird ohne weitere Interaktion die Alternative ausgewählt, für die die höchste Priorität vergeben wurde. Die Priorität legt der Paketmaintainer fest. Für manche Gruppen von Alternativen gibt es jedoch feste Regeln zur Berechnung der Prioritäten, so z.B. für Window-Manager. Diese sind in Abschnitt 11.8.4 des Debian Policy Manuals festgelegt [Debian-Policy-Manual]. Installieren Sie bspw. `vim-gtk` auf einem System, auf dem bisher `nano` der Editor mit der höchsten Priorität war, so werden bspw. die Datei `/usr/bin/editor` und `/etc/alternatives/editor` automatisch auf die grafische Variante von Vim umgestellt.

Hinweise über die automatische Auswahl von Alternativen bei der Paketinstallation

```
[...]
Setting up vim-gtk (2:7.4.488-4) ...
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/vim (vim) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/vimdiff (vimdiff) in auto ↩
    mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/rvim (rvim) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/rview (rview) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/ex (ex) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/editor (editor) in auto ↩
    mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/gvim (gvim) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/gview (gview) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/rgview (rgview) in auto ↩
    mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/rgvim (rgvim) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/evim (evim) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/eview (eview) in auto mode
update-alternatives: using /usr/bin/vim.gtk to provide /usr/bin/gvimdiff (gvimdiff) in auto ↩
     mode
[...]
```

Die präferierte Alternative für ein gegebenes generisches Programm ändern Sie mit der Option `--config`. Dabei entscheiden Sie auch, ob bei zukünftigen Paketinstallationen die von Ihnen präferierte Alternative automatisch neu ausgewählt werden soll, oder ob die manuell ausgewählte Alternative stets beibehalten werden soll. Damit behalten Sie die derzeit ausgewählte Alternative unverändert bei.

Ändern des systemweiten Standardeditors von einer automatischen Wahl auf zile

```
$ update-alternatives --config editor
Es gibt 10 Auswahlmöglichkeiten für die Alternative editor (welche
/usr/bin/editor bereitstellen).

  Auswahl      Pfad              Priorität Status
- ------------------------------------------------------------
* 0            /usr/bin/vim.gtk   50        automatischer Modus
  1            /bin/ed           -100       manueller Modus
  2            /bin/elvis-tiny    10        manueller Modus
  3            /bin/nano          40        manueller Modus
  4            /usr/bin/emacs24   0         manueller Modus
  5            /usr/bin/mcedit    25        manueller Modus
  6            /usr/bin/nvi       19        manueller Modus
  7            /usr/bin/vigor    -150       manueller Modus
  8            /usr/bin/vim.gtk   50        manueller Modus
  9            /usr/bin/vim.nox   40        manueller Modus
  10           /usr/bin/zile      30        manueller Modus

Drücken Sie die Eingabetaste, um die aktuelle Wahl[*] beizubehalten,
oder geben Sie die Auswahlnummer ein: 10
update-alternatives: /usr/bin/zile wird verwendet, um /usr/bin/editor
(editor) im manuellen Modus bereitzustellen
$
```

Bei manchen Paketen wurde dem Prioritätswert mit einem Augenzwinkern sogar noch eine zusätzliche Bedeutung untergeschoben. So zeigen zum Beispiel die Prioritätswerte für die deutschsprachigen Wörterbücher aus den Paketen *aspell-de* und *aspell-de-alt* gleichzeitig auch das Jahr an, in welchem die entsprechende Reform der Rechtschreibung in Kraft trat.

Beispiel mit viel Humor in den deutschsprachigen aspell-Wörterbüchern

```
There are 2 choices for the alternative de.multi (providing /usr/lib/aspell/de.multi).

  Selection    Path                          Priority   Status
- ------------------------------------------------------------
```

```
  0            /usr/lib/aspell/de-neu.multi   1996      auto mode
* 1            /usr/lib/aspell/de-alt.multi   1901      manual mode
  2            /usr/lib/aspell/de-neu.multi   1996      manual mode
```

Kapitel 19

Backports

```
überarbeiteter Beitrag aus aus: http://www.wizards-of-foss.de/de/weblog/2013/03/08/debian- ↩
    backports-verwenden/
```

19.1 Ausgangssituation

Debian handhabt die Pakete seiner stabilen Veröffentlichung *stable* (siehe Abschnitt 2.10) äußerst konservativ und lässt (mit extrem wenig Ausnahmen) keinerlei neue Funktionalitäten zu. Die einzigen vorgesehenen Aktualisierungen in *stable* sind Sicherheitsupdates oder die Behebung gröberer Bugs.

Benötigen Sie dennoch von einzelnen Programmen neuere Versionen — sei es wegen neuerer Funktionen oder aufgrund von Kundenanforderungen — so gibt es verschiedene Wege, diese dennoch über das Paketsystem zu bekommen:

1. Die Anwendung lokal kompilieren und nach `/usr/local` installieren,
2. Nutzen der Entwicklungszweige *testing* oder *unstable*,
3. Pakete aus den Entwicklungszweigen *testing* oder *unstable* auf *stable* installieren,
4. Pakete aus den Entwicklungszweigen *testing* oder *unstable* auf *stable* neu bauen, oder
5. Pakete aus dem Backports-Repository verwenden, falls diese darin verfügbar sind.

19.2 Gegenüberstellung der verschiedenen Lösungsansätze

Der erste Lösungsansatz — das lokale Kompilieren und Installieren — geht am Paketsystem vorbei und Sie verlieren damit sämtliche Vorteile der Paketverwaltung. Handelt es sich um ein Programm, von dem andere Pakete abhängen, kommt noch das Problem hinzu, dass diese Abhängigkeiten dann im Paketsystem nicht mehr erfüllt sein müssen. Wir gehen daher auf diese Option nicht näher ein, auch wenn es sicher Fälle gibt, in denen dies nicht die schlechteste Variante ist.

Das Nutzen von Debians *testing*- oder *unstable*-Zweigen anstatt *stable* bedingt, dass nicht nur die benötigte Software, sondern das komplette Betriebssystem mit allen Anwendungen in einer neueren Version verwendet wird. Dabei nehmen Sie das Risiko in Kauf, dass die Pakete noch nicht die Stabilitätskriterien von Debian erfüllen und daher durchaus Fehler, bei *unstable* sogar Inkonsistenzen und Uninstallierbarkeit einzelner Pakete auftreten können. Dazu kommt, dass sich Pakete aus *testing* und *unstable* im dauernden Fluß befinden und sich daher relativ oft bezüglich der Konfigurationsformate oder ihrer unterstützten Funktionen ändern — wie bei einem sogenannten *Rolling Release*.

Manchmal, aber lange nicht immer, sind die Abhängigkeiten eines Pakets aus *testing* oder *unstable* nicht allzu restriktiv und Sie können dieses — z.B. mit dem Aufruf `dpkg -i` — einfach so auch auf einem *stable*-System installieren. Das bedeutet aber nicht zwangsläufig, dass die nächste Version dieses Pakets aus Debians Entwicklungszweigen dies immer noch tut. Sie gehen damit das Risiko ein, das Paket nicht aktualisieren zu können oder es weiter benutzen zu können.

Um nicht auf die Stabilität und die Sicherheitsupdates von Debians *stable*-Zweig zu verzichten und gleichzeitig einzelne Anwendungen in neueren Versionen nutzen zu können, ist der sauberste Weg die Rückportierung (engl. *to backport*). Die Grundlage stellen die Versionszweige *testing*, *unstable* und auch *experimental* dar. Damit erreichen Sie, dass sich das ausgewählte Paket auch auf dem *stable*-Zweig bauen und installieren lässt. Es gehört meistens eine kleine Portion Glück dazu, das Paket ohne Murren zu portieren. Solche Backports sind nicht immer trivial in der Entwicklung und können auch vergleichsweise aufwendig in der Pflege sein, z.B. wenn neuere Versionen weitere Backports benötigen, weil in der nachfolgenden Entwicklung weitere Abhängigkeiten hinzukamen oder neuere Versionen der referenzierten Abhängigkeiten benötigt werden.

19.3 Debian Backports

Dass o.g. Bedürfnisse, Gedanken und Lösungen nichts Neues sind, können Sie sich sicher denken. Deswegen hat Norbert Tretowski 2003 *backports.org*, kurz *bpo*, ins Leben gerufen — einen zentralen Platz für solche Rückportierungen.

Bedarf und Interesse an den Backports wuchs und es gab ein eigenes Spiegelnetzwerk analog zu dem von Debian. 2010 wurde das Projekt dann mit dem Umzug von *backports.org* zu *backports.debian.org* offizieller Bestandteil des Debian-Projektes [backports.org-moved-to-backports.debian.org].

Dennoch wurde die von nun an *Debian Backports* genannten Pakete noch eine Weile getrennt von den offiziellen Paketen der Distribution verteilt. Erst seit Debian 7 *Wheezy* sind die Backports ebenfalls im selben Paketpool enthalten, jedoch nach wie vor in einem eigenen Zweig, der auf die Zeichenkette `-backports` endet. Aus diesem Grunde sind Backports für Debian 6 *Squeeze* leicht anders einzurichten, als für darauf folgende Veröffentlichungen.

19.4 Welche Pakete gibt es als offiziellen Backport?

Welche Pakete es auch als offiziellen Backport gibt, können Sie mittlerweile recht einfach in der Debian Paketsuche [Debian-Paketsuche] herausfinden. Ob es ein solches Paket auch als offiziellen Backport gibt, hängt von zwei Dingen ab — dem Bedarf (wesentliche Änderungen gegenüber der Version im *stable*-Zweig) und jemanden, der den Backport initial macht und dann auch weiterhin pflegt.

Letzteres muss nicht der eigentliche Paketbetreuer des Pakets in Debian sein. Es kommt durchaus vor, dass dieser kein Bedarf bzw. Interesse an einem Backport hat. In diesem Fall ist es nicht unüblich, dass sich jemand anderes um den (nach wie vor offiziellen) Backport kümmert. U.a. aus diesem Grund sollten Sie Bug-Reports gegen Pakete aus den Debian Backports stets an die Backports-Mailingliste [Backports-Mailingliste] senden und nicht an das normale Bug-Tracking-System von Debian (Stand Januar 2015).

19.5 Welche Versionen gibt es als offizielle Backports?

Für den Zweig *stable-backports* der Debian Backports sind nur Versionen erlaubt, die momentan in Debians *testing*-Zweig enthalten sind. Für den *oldstable-backports*-Zweig — quasi als Backports für die vorletzte *stable*-Veröffentlichung von Debian — sind nur Paketversionen aus der aktuellen *stable*-Veröffentlichung zugelassen. Diese Varianten haben die Bezeichnung *oldstable*. Hintergrund für diese Zuordnung ist, dass es möglich sein soll, bei der Aktualisierung eines Systems mit (offiziellen) Backports auf die jeweils nächste *stable*-Veröffentlichung alle bestehenden Backports automatisch durch die entsprechenden, neuen *stable*-Pakete zu ersetzen.

Trotzdem gibt es auch hier immer wieder Bedarf für Backports von *testing* nach *oldstable*, die von o.g. Regel des sauberen Upgrades abweichen. Aus diesem Grund gibt es für die *oldstable*-Veröffentlichungen von Debian neben den *oldstable-backports* auch noch *oldstable-backports-sloppy*. Das englische Wort *sloppy* steht für „schlampig“, „schluderig“ oder „nachlässig“ und besagt, dass diese Backports o.g. Anforderung an eine saubere Aktualisierbarkeit auf die nächste *stable*-Veröffentlichung nicht entsprechen.

19.6 Einbindung in den Paketbestand

Backports sind nicht von Hause aus aktiviert und Sie müssen diese in der Paketverwaltung explizit ergänzen. Dazu fügen Sie in der Liste der Paketquellen unter `/etc/apt/sources.list` (siehe dazu Abschnitt 3.3) einen entsprechenden Eintrag für die

passenden Backports ihrer Distribution hinzu. Für Debian 7 *Wheezy* und die Distributionsbereiche *main*, *contrib* und *non-free* sieht der Eintrag wie folgt aus:

Eintrag zu Debian Backports für Debian 7 Wheezy

```
# Backports
deb http://httpredir.debian.org/debian wheezy-backports main contrib non-free
```

Wie bereits oben erwähnt, findet sich das APT-Repository für die Backports für Debian 6 *Squeeze* in einem getrennten Spiegelnetzwerk. Deswegen ist dort die kanonische Mirror-Adresse eine leicht andere:

Eintrag zu Debian Backports für Debian 6 Squeeze

```
# Backports
deb http://httpredir.debian.org/debian-backports squeeze-backports main contrib non-free
```

Nach der Aktualisierung der Paketquellen — bspw. mittels `apt-get update` — stehen Ihnen die zusätzlichen Pakete bereits zur Verfügung. Diese werden jedoch nicht automatisch berücksichtigt und installiert, sondern dazu bedarf es noch eines expliziten Aufrufs mit zusätzlichen Schaltern. `apt-get` und `aptitude` benutzen dazu den Schalter `-t` gefolgt vom Namen des Backports-Archivs.

Um bspw. das Paket *asciidoc* aus den Backports für Debian 7 *Wheezy* nachzuziehen, geben Sie auf der Kommandozeile folgendes ein:

Installation eines Pakets mit expliziter Angabe der Distribution wheezy-backports

```
# apt-get -t wheezy-backports install asciidoc
...
#
```

Alternativ können Sie auch mittels Pinning paketweise bestimmen (siehe Abschnitt 20.4), bei welchen Paketen Backports verwendet werden sollen.

19.7 Weiterführende Dokumentation

Die offizielle Dokumentation auf Englisch gibt es auf der Backports-Projektseite [Debian-Backports]. Eine deutschsprachige Anleitung finden Sie im Wiki von debianforum.de [Debianforum-Wiki-Backports].

19.8 Backports bei Ubuntu

Auch bei Ubuntu gibt es Backports. Diese funktionieren nach ähnlichen Regeln wie bei Debian. Da es bei Ubuntu aber keinen *testing*-Zweig wie bei Debian gibt und die Veröffentlichungen wesentlich häufiger passieren, werden Backports dort üblicherweise von der aktuellen Veröffentlichung zur vorherigen Veröffentlichung oder zur vorherigen LTS-Veröffentlichung gemacht.

19.9 Wichtige Fragen, die sich bei Backports ergeben

- wie kommt ein Backport-Paket zustande? Sicher gibt es dazu einen definierten Arbeitsablauf
- Laufen die Pakete außerhalb des üblichen Validierungsprozesses (ähnlich wie Ubuntu PPAs)
- wann ist die Installation eines Backport-Pakets sinnvoll, wann nicht?
- kann bei Backports was schiefgehen? Wenn ja, was? Kann ich das vorher irgendwie testen?
- Gibt es Updates dazu? Pflege ich die über den üblichen `apt-get update`-Prozess ein, oder geht das anders, bspw. manuell?
  - Pakete werden wie ein normales Paket ausgewählt und gepflegt

- Wie entferne ich ein Backport-Paket wieder (`apt-get remove Paketname`)?
  - ja
- Oder meintest Du "Wie downgrade ich ein Backport-Paket wieder?"

Kapitel 20

Veröffentlichungen mischen

Debian reglementiert die Erweiterung der vorhandenen Funktionalität für die Veröffentlichung eines Pakets aus Debian *stable* deutlich. Besteht Ihrerseits jedoch die Notwendigkeit für eine neuere Paketversion mit zusätzlichen Funktionen, prüfen Sie als erstes, ob es für Ihr Paket bereits einen Kandidaten aus dem Bereich *Debian Backports* gibt (siehe Kapitel 19). Ein solches Paket ist dann auch auf Debian *stable* zugeschnitten.

Bleibt diese Suche erfolglos, ist der nächste Schritt die Recherche nach neueren Paketen in den Veröffentlichungen Debian *testing* und *unstable* sowie das nachfolgende Mischen der unterschiedlichen Veröffentlichungen. Dafür bestehen zwei Wege — die Paketauswahl mit der expliziten Angabe der Veröffentlichung (siehe Abschnitt 20.2) und der paketweisen Festlegung von Prioritäten, genannt APT-Pinning (siehe Abschnitt 20.4).

Bei beiden Wegen werden zwei Dinge angestrebt — einerseits eine neuere Programmversion einzuspielen, ohne eine vollständige Aktualisierung Ihrer gesamten Installation durchführen zu müssen (siehe Abschnitt 8.45), und andererseits das Gesamtsystem möglichst stabil zu halten. Das ganze ist ein Balanceakt ohne die Garantie von Debian *stable*.

Problematisch ist dabei, dass Abhängigkeiten zwischen den verschiedenen Paketversionen aus den genutzten Veröffentlichungen bestehen. Aus der Erfahrung heraus wissen wir jedoch, dass das ganze zwischen zwei aufeinanderfolgenden Veröffentlichungen weitestgehend reibungslos funktioniert, bspw. aus der Kombination von Debian *stable* und *testing*. Der Hintergrund liegt darin, dass in diesem Fall die Unterschiede zwischen den Paketen aus den Veröffentlichungen noch nicht ganz so groß sind ([Jurzik-Debian-Handbuch], [Drilling-APT-Pinning-LinuxUser]). Sollte Ihnen das Ergebnis nach dem APT-Pinning dennoch zu instabil sein, bleibt als Folgeschritt stets noch ein Distributionsupgrade übrig (siehe Abschnitt 8.45).

20.1 Die bevorzugte Veröffentlichung für alle Pakete festlegen

Hilfreich ist die Festlegung einer bevorzugten Veröffentlichung — einer sogenannten *target release*. Daran orientiert sich APT und benutzt nur Pakete dieser Veröffentlichung — egal, was sonst noch an anderen Paketversionen existiert.

APT entnimmt die Veröffentlichung der Datei `/etc/apt/apt.conf`. Sofern diese Datei noch nicht vorhanden ist, legen Sie sie an. In die Datei tragen Sie die gewünschte Veröffentlichung ein, bspw. Debian *stable* wie folgt:

Debian stable als bevorzugte Veröffentlichung festlegen

```
APT::Default-Release "stable";
```

20.2 `apt-get` mit expliziter Angabe der Veröffentlichung

APT richtet sich nach den festen Einstellungen in den Konfigurationsdateien. Mit dem Schalter `-t` (Langform `--target-release` oder `--default-release`) übergehen Sie diese Festlegung und legen explizit fest, aus welcher Veröffentlichung Sie das Paket beziehen möchten. Alternativ geben Sie die Veröffentlichung ohne den Schalter und direkt nach dem Paketnamen an.

Für das Paket *gdm3* und die Veröffentlichung Debian *testing* sind die folgenden beiden Aufrufe zulässig:

Aufrufe mit expliziter Angabe der Veröffentlichung

```
apt-get -t testing install gdm3
apt-get install gdm3/testing
```

Um dieses Paket einzuspielen, erzeugt APT eine Vorgabe-Pin mit der Priorität 990 (siehe dazu Tabelle 20.1). Daher wird das Paket installiert, es sei denn, es gibt bereits eine Version, die zur festgelegten Zielveröffentlichung gehört oder die bereits vorhandene Version des Pakets ist neuer als das benannte Paket.

20.3 Von APT zu APT-Pinning

Bisher wurden im Buch alle Pakete gleichwertig behandelt. Zusätzlich bekommen die Pakete nun unterschiedliche Prioritäten. Die Paketverwaltung mit APT überprüft verschiedene „Schlüsselstellen“ und klärt, was für das jeweilige Paket eingestellt ist. Daraus ermittelt APT für jedes Paket dessen Priorität und wählt für die Installation die Version mit der höchsten „Vorrangstufe“ aus. Liegt keine explizite Angabe vor, kommen die Standardeinstellungen zum Tragen (siehe auch Abschnitt 8.18).

Die Steuerung der Vorrangstufe erfolgt über einzelne Zahlenwerte, genannt *Pins*. Anhand dieser wird ausgewählt, aus welchen Paketquellen die Pakete bezogen werden. Das ganze kann beliebig verkompliziert werden, da APT über den Zahlenwert entscheidet, was es installiert, nicht installiert oder aktualisiert (siehe dazu Tabelle 20.1).

Tabelle 20.1: Verwendete Prioritäten beim APT-Pinning

Priorität	**Bedeutung**
unter 0	Das Paket wird niemals installiert.
zwischen 0 und 100	Das Paket wird nur dann installiert, wenn noch keine Version davon auf dem System installiert ist.
zwischen 100 und 500	Das Paket wird installiert, es sei denn, es gibt eine Version, die zu einer anderen Veröffentlichung gehört, oder die bereits vorhandene Version ist neuer.
zwischen 500 und 990	Das Paket wird installiert, es sei denn, es gibt eine Version, die zur festgelegten Zielveröffentlichung gehört, oder die bereits vorhandene Version ist neuer.
zwischen 991 und 1000	Das Paket wird immer installiert, es sei denn, die bereits vorhandene Version ist neuer.
Wert größer als 1000	Das Paket wird immer installiert, auch wenn das ein Downgrade auf eine ältere Version bedeutet.

20.4 Paketweise festlegen

Für diesen Fall besteht eine Liste mit Einträgen für einzelne Pakete und ganze Paketgruppen. Bis Debian 6 *Squeeze* beinhaltete die Datei `/etc/apt/preferences/` die Prioritäten für alle Paketnamen, Veröffentlichungen, Hersteller und Versionen.

Ab Debian 7 *Wheezy* wurde das in das Verzeichnis `/etc/apt/preferences.d/` ausgelagert. Jede Datei in diesem Verzeichnis darf beliebig viele Festlegungen beinhalten, wobei der Dateiname jeweils frei wählbar ist. Die Abarbeitung der einzelnen Einträge erfolgt von oben nach unten, wobei nachfolgende, mehrfache Einträge ignoriert werden. Jeder Eintrag, d.h. jede Festlegung, besteht aus den folgenden drei Zeilen:

Eintrag für ein Paket

```
Package: *
Pin: release a=stable
Pin-Priority: 50
```

Obiger Eintrag besagt, dass APT nur Pakete aus dem Bereich Debian *stable* und nicht aus Debian *testing* oder *unstable* installiert. Dabei stehen die einzelnen Schlüsselworte jeweils für:

`Package`
Paketname, für welches die Zuordnung gilt. Ein `*` bezeichnet alle Pakete.

`Pin`
Nach dem Schlüsselwort `release` spezifizieren Sie die Veröffentlichung (siehe Abschnitt 2.10). Dabei ist hier die Angabe eines Aliasnamens wie *Wheezy* oder *Sid* nicht erlaubt. Zulässig sind aber bspw. die Versionsnummer, der Distributionsbereich und die Herkunft. Eine genaue Auflistung enthält Tabelle 20.2.

`Pin-Priority`
Das bezeichnet den Zahlenwert für die Pin. Welche Werte zulässig sind, entnehmen Sie bitte Tabelle 20.1 in Abschnitt 20.3.

Tabelle 20.2: Zulässige Parameter beim APT-Pinning

Parameter und Schlüsselwort	Bedeutung	Beispiel
`a` (`archive`)	Veröffentlichung (siehe Abschnitt 2.10)	*unstable*
`c` (`component`)	Distributionsbereich (siehe Abschnitt 2.9)	*main*
`l` (`label`)	Bezeichner	Debian
`n` (`name`)	Aliasnamen der Veröffentlichung (siehe Abschnitt 2.10.2)	*Jessie*
`o` (`origin`)	Herkunft	Debian
`v` (`version`)	explizite Versionsnummer (siehe Abschnitt 2.11)	6.0.3

20.5 Praktische Beispiele

Anhand von drei typischen Einträgen verdeutlichen wir Ihnen nachfolgend, wie die Einträge für ein erfolgreiches APT-Pinning aussehen müssen.

In **Beispiel 1** legen Sie eine bestimmte Veröffentlichung fest. Alle Pakete kommen aus Debian 7.5 *Wheezy* und werden durch die Pin mit dem Wert 1000 gegen eine unbeabsichtigte, automatische Entfernung geschützt.

Beispiel 1: Veröffentlichung festlegen

```
Package: *
Pin: release v=7.5, l=Debian
Pin-Priority: 1000
```

In **Beispiel 2** legen Sie fest, dass ein Paket in einer bestimmten Version gehalten wird. Die Angabe `samba*` bezieht sich hier auf alle Debianpakete, die `samba` im Paketnamen tragen. Die Angabe `v=3.5.6*` bewirkt, dass diese Pakete in der Version 3.5.6 erhalten bleiben, d.h. nicht aktualisiert werden. Durch die Pin mit dem Wert 1000 werden die Pakete zudem gegen eine unbeabsichtigte, automatische Entfernung geschützt.

Beispiel 2: Paket in ausgesuchter Version halten

```
Package: samba*
Pin: v=3.5.6*
Pin-Priority: 1000
```

In **Beispiel 3** legen Sie die Paketherkunft genauer fest. Das gilt nur für die Pakete aus der Gruppe `gnome`, die zudem von der URL ftp.informatik.tu-berlin.de stammen. Durch die Pin mit dem Wert 600 gilt das nur, sofern diese Pakete aktueller als die bisherigen Pakete sind.

Beispiel 3: Paketherkunft bestimmten

```
Package: *gnome
Pin: origin ftp.informatik.tu-berlin.de
Pin-Priority: 600
```

Kapitel 21

Pakete bauen mit `checkinstall`

21.1 Pakete aus zusätzlichen Quellen ergänzen

Das Debian-Paketarchiv ist bereits sehr umfangreich und umfaßt eine Vielzahl von vorab geprüften Paketen mit ausgewählter, stabiler Software. Benötigen Sie hingegen Softwarekomponenten, welche nicht darin enthalten ist — aus welchem Grund auch immer --, können Sie beispielsweise auf zusätzliche Quellen für Fremdpakete zurückgreifen und die entsprechenden Pakete daraus einbinden (siehe dazu „Paketquellen“ in Abschnitt 3.1).

21.2 Software selbst übersetzen und einspielen

Als Alternative zu obigem Weg stehen Ihnen stets die entsprechenden Sourcepakete oder der Quelltext als `tar.gz`-Archiv von der Projektseite zur Verfügung. In beiden Fällen übersetzen Sie den Programmcode selbst und installieren danach die dabei erzeugten Binärdateien auf ihrem System. Üblicherweise umfaßt das den Dreierschritt aus den Aufrufen `./configure`, `make` und `make install`.

Das geht jedoch an der Paketverwaltung vorbei — diese bemerkt nicht, daß Sie ihrem System zusätzliche Software hinzugefügt haben. Dieser Schritt birgt die Risiken, daß a) nicht alle Abhängigkeiten der Software erfüllt werden, b) Konflikte mit einer gleichzeitig installierten, anderen Version des Programms entstehen und Sie c) die Software nur sehr mühselig wieder von ihrem System entfernen können. Nicht wenige Entwickler „vergessen“ in ihren Makefiles das Ziel `uninstall` (siehe [Drilling-Checkinstall-LinuxUser]).

Gleiches Ungemach droht Ihnen, wenn Sie zu einem späteren Zeitpunkt die eingespielte Software oder auch andere Pakete aktualisieren möchten. Nicht selten treten dann erhebliche Abhängigkeitsprobleme zu anderen Paketen auf. Komponenten, die Sie an der Paketverwaltung vorbei installiert haben, sind zwar vorhanden, werden aber von dieser als fehlende Abhängigkeit eingestuft.

21.3 Software selbst übersetzen und als `deb`-Paket einspielen

An dieser Stelle kommt das Projekt `checkinstall` [checkinstall] ins Spiel. Es steht unter der GPLv2 und ist über das gleichnamige Paket aus den Repositories verfügbar [Debian-Paket-checkinstall].

Anmerkung

Handelt es sich bei der einzuspielenden Software um Perl-Module, hilft Ihnen auch das spezialisierte Paket *dh-make-perl* weiter [Debian-Paket-dh-make-perl]. Wie Sie das Werkzeug verwenden, erläutert Steve Kemp in seinem lesenswerten Blogeintrag „Building Debian packages of Perl modules“ [Kemp-dh-make-perl].

`checkinstall` hat sich zum Ziel gesetzt, das Übersetzen von Paketen aus dem Quellcode und das direkte Erstellen von Binärpaketen miteinander zu kombinieren. Diese „frischen“ Binärpakete passen dann exklusiv zu Ihrer bestehenden Installation. Es kann neben `deb`-Paketen auch `rpm`- und `tgz`-Dateien für Slackware erstellen. Die einzelnen Schalter entnehmen Sie bitte Tabelle 21.1. Benennen Sie keinen Schalter, erzeugt `checkinstall` automatisch ein `deb`-Paket.

Tabelle 21.1: Schalter für die verschiedenen unterstützten Paketformate

Paketformat	**Schalter**
Debian-Paket (`deb`)	`-D`
	`-t debian`
	`--type=debian`
RPM-Paket (`rpm`)	`-R`
	`-t rpm`
	`--type=rpm`
Slackware-Paket (`tgz`)	`-S`
	`-t slackware`
	`--type=slackware`

Vereinfacht gesagt, erstellt `checkinstall` ein Paket direkt aus dem Quellcode und übergeht den Paketmanager dabei aber nicht. Es beobachtet den Erstellprozess und bindet alle Dateien in das neue Paket ein, die bei der Übersetzung entstehen und benötigt werden. Daher gibt es auch dafür die schöne Umschreibung „Installations-Verfolger“.

Dazu bezieht `checkinstall` zunächst die benötigten Quellen zum Paket. Als Quelle kommen alle Softwarearchive in Frage, so z.B. neben den regulären Debian-Paketquellen auch von den Plattformen SourceForge [SourceForge], Freecode (vormals Freshmeat) [FreeCode] und GitHub [GitHub]. Gleiches gilt für das direkte Auschecken aus dem Versionskontrollsystem des Projektes.

Zum Aufruf genügt das nachfolgende Kommando im Verzeichnis mit dem Quellcode. Es entspricht dem bereits oben genannten Aufruf von `./configure`, `make` und `make install` und ist gleichzeitig die Kurzform für den Aufruf `checkinstall --install=yes`.

```
# checkinstall
```

Aus dem zunächst bezogenen `tar.gz`-Archiv baut `checkinstall` ein zu ihrer Installation passendes `deb`-Paket und installiert dieses über die Paketverwaltung. Dabei erhält das Paket die Markierung *hold* (siehe dazu Abschnitt 2.15).

Möchten Sie ein bestimmtes Skript zur Installation ausführen, geben Sie dieses beim Aufruf von `checkinstall` als zusätzlichen Parameter an. Nachfolgend heißt das Skript schlicht `installationsskript.sh`, kann aber von Ihnen beliebig benannt werden.

```
# checkinstall installationsskript.sh
```

Wünschen Sie hingegen keine automatische Installation, rufen Sie `checkinstall` mit dem Parameter `--install=no` auf. Das entspricht den beiden Aufrufen `./configure` und `make`.

```
# checkinstall --install=no
```

Weitere Debian-spezifische Schalter entnehmen Sie bitte Tabelle 21.2. Diese Schalter korrespondieren direkt mit den dazugehörigen Feldern in einem Debian-Binärpaket (siehe Abschnitt 4.1). Schalter zur Darstellung und Ausgabe entnehmen Sie bitte der Manpage zum Programm oder über den Aufruf von:

```
checkinstall --help
```

Tabelle 21.2: Spezifische Schalter für ein Debian-Binärpaket

Schalter	Bedeutung
`--pkgname=`*name*	Name des Pakets
`--pkgversion=`*version*	Versionsnummer des Pakets
`-A` *Architektur*	Architektur des Pakets
`--arch` *Architektur*	Architektur des Pakets
`--pkgarch=`*Architektur*	Architektur des Pakets
`--pkgrelease=`*Release*	Angabe der Veröffentlichung
`--pkglicense=`*Lizenz*	Angabe der Lizenz zum Paket
`--pkggroup=`*Gruppe*	Benennung der Paketkategorie
`--pkgsource=`*Quelle*	Angabe der Quelle zum Paket
`--pkgaltsource=`*Quelle*	alternative Angabe der Quelle zum Paket
`--pakdir=`*Verzeichnis*	Zielverzeichnis, in dem das Paket gespeichert wird
`--maintainer=`*Emailadresse*	Emailadresse des Paketmaintainers
`--provides=`*Liste*	Name der Pakete, die es bereitstellt
`--requires=`*Liste*	Name der Pakete, die das Paket benötigt
`--conflicts=`*Liste*	andere Pakete, mit denen das Paket in Konflikt steht
`--replaces=`*Liste*	andere Pakete, die dieses Paket ersetzt
`--dpkgflags=`*Flags*	Flags, die an `dpkg` zur Installation mitgegeben werden
`--nodoc`	keine Dokumentation in das Paket einfügen

21.4 Beispiel

ToDo

21.5 Vor- und Nachteile

Erwartungsgemäß wird die Verwendung von `checkinstall` im Alltag von Entwicklern, Paketmaintainern und Benutzern gemischt bewertet.

Entwickler und *Paketmaintainer* sehen das Werkzeug überwiegend positiv — haben Sie darüber nämlich die Möglichkeit, auszutesten, ob sich die von Ihnen verwendete Entwicklerversion nahtlos in das bestehende Debian-Ökosystem einspielen läßt und mit den anderen Paketen harmoniert. Die Paketverwaltung hat eine Information darüber, daß Sie ein zusätzliches Paket als solches installiert haben. Sollte das Paket unerwartete Querschläger produzieren, kratzen Sie es auch wieder ohne viel Aufwand und vorallem restefrei vom System herunter. Das erlaubt Ihnen automatisierte Integrationstests im Rahmen der Qualitätssicherung.

Verschwiegen werden sollen jedoch nicht die Nachteile. Ein mittels `checkinstall` generiertes und eingespieltes Paket unterläuft die strenge Qualitätskontrolle von Debian. Die übliche Validierung durch andere Tester und Benutzer sowie eine Erzeugung von Prüfsummen findet ebenfalls nicht statt. Die Paketabhängigkeiten werden nur bedingt validiert, d.h. nur auf dem System, auf dem das Paket erzeugt und eingespielt wurde. Das Ergebnis — sprich das erzeugte `deb`-Paket — ist nicht unbedingt portabel und stets eins-zu-eins auf andere Debian-Installationen übertragbar. Berichtet wird ebenfalls, daß `checkinstall` nicht bei Programmen funktioniert, die statisch gegen die `libc` linken, oder bei solchen, bei denen das SUID/GUID-Bit gesetzt ist (siehe [Drilling-Checkinstall-LinuxUser]). Ebenso sind Probleme mit den `preinstall`- und `postinstall`-Skripten bekannt (siehe [Schnober-Checkinstall-LinuxUser]).

Aus Sicht der *Benutzer* ist es sicherlich sehr erfreulich, wenn eine Lücke in den benötigten Komponenten geschlossen wird. Was sie meist weniger einschätzen können, ist die Stabilität der Lösung und der Aufwand seitens der Entwickler und Paketmaintainer, um diese Lösung dauerhaft zu betreuen und den Weiterentwicklungen anzupassen.

Aus den oben genannten Gründen empfehlen wir Ihnen, den Einsatz von `checkinstall` genau zu überdenken. Als nützlich und hilfreich schätzen wir es insofern ein, daß Sie in einem Schritt überprüfen können, ob ein Stapel Software „baut“, sich ein Paket daraus schneidern läßt und dieses mit dem restlichen Debian-Ökosystem zusammenarbeitet. Das erleichtert Ihnen die Integrationstests und sollte nachfolgend die Basis dafür bilden, daraus ein richtiges Paket entsprechend den Debian-Paket-Standards

zu erstellen. Bis das soweit ist, steht einer Benutzung auf lokalen Rechnern und Einzelsystemen — bspw. im Rahmen einer „Testinstallation“ — nichts im Wege. Einem großflächigen Alltagsbetrieb des erzeugten Pakets stehen wir skeptisch gegenüber.

Interessant ist die Verwendung von `checkinstall` in Kombination mit dem Werkzeug `auto-apt` [Debian-Paket-auto-apt]. Damit fügen Sie fehlende Pakete bei Bedarf hinzu und lösen offene Paketabhängigkeiten auf. Genauer gehen wir drauf unter „Fehlende Pakete bei Bedarf hinzufügen“ in Kapitel 17 ein.

21.5.1 Weitere noch unbearbeitete Notizen

- Vorteile:
  - installiert automatisch die zusätzlichen, bisher noch nicht installierten Header-Files nach
    - stimmt das?

Kapitel 22

Paketformate mischen

22.1 Einführung

Debian GNU/Linux und seine Derivate setzen auf das `deb`-Format auf. Eine Vielzahl Pakete stehen in diesem Format bereit und erlauben die Zusammenstellung und den Betrieb stabiler Systeme. Mitunter treten Situationen auf, die die Einbindung weiterer Software erfordern, die in einem anderen Paketformat vorliegt, bspw. `rpm`. Die Gründe dafür sind vielfältig:

- Die Software wurde nur zusammengestellt und liegt bislang nur als `tar.gz`-Archiv vor.
- Die Software ist bislang nicht anders paketiert, weil sich bspw. der Entwickler nur mit genau diesem Paketformat und dem Mechanismus zur Paketierung auskennt.
- Das bestehende `deb`-Paket liegt zu weit zurück („ist zu alt“) und neuere Features werden benötigt. Eine neuere Variante ist jedoch in einem anderen Paketformat erhältlich.
- Das gewünschte Paket oder die benötigte Version wurde noch nicht in die stabile Veröffentlichung aufgenommen. Das Paket ist noch zu neu und liegt „in Quarantäne“.
- Die Software bzw. das Debianpaket wurde noch nicht für ihre gewünschte Plattform portiert.

Helfen Ihnen an dieser Stelle *Debian Backports* (siehe Kapitel 19) oder das Mischen von Veröffentlichungen (siehe Kapitel 20 nicht weiter, stellt die Verwendung eines Pakets im Fremdformat eine Variante zur Lösung dar. Nachfolgend gehen wir darauf ein, wie Ihnen dabei das Programm `alien` helfen kann (siehe Abschnitt 22.2).

22.2 Fremdformate mit `alien` hinzufügen

22.2.1 Einführung

Den Begriff *alien* übersetzen Sie am ehesten mit *fremd*, *Fremdling* oder *Ausländer*. So heißt auch das gleichnamige Debianpaket [Debian-Paket-alien], welches Ihnen hilft, eine Software, welche nicht als `deb`-Paket vorliegt, entsprechend umzuwandeln und für Ihre Debian-Installation vorzubereiten.

Wie bereits in „Gestaltwandler. Programmpakete richtig konvertieren“ [Hofmann-Osterried-Alien-LinuxUser] beschrieben, sollten bei der Umwandlung des Pakets nach Möglichkeit die folgenden Bestandteile erhalten bleiben:

- die Paketbeschreibungen. Damit erkennen Sie später über die Paketverwaltung, um was für eine Software es sich handelt, wer der Autor ist oder wo sich die dazugehörige Homepage des Projekts befindet.
- die Informationsdateien. Diese beschreiben, wie Sie das Paket wieder entfernen oder aktualisieren. Häufig befinden sich auch Hinweise dabei, die benennen, was dabei gegebenenfalls zu beachten ist.

- die Informationen über Abhängigkeiten zu anderer Software.
- die Angabe der Prozessorarchitektur im Dateinamen, wie etwa *amd64* bei `deb`-Paketen und *x86_64* bei `rpm`-Archiven.

Bei der Umwandlung des Paketformats bestehen mehrere Fallstricke. Diese führen regelmäßig dazu, dass die gewünschte Software nicht wie erhofft funktioniert:

- die referenzierten Bibliotheken passen gar nicht, nicht mehr (sind bspw. veraltet) oder sind nicht auffindbar (sind bspw. nicht installiert oder haben einen anderen Paketnamen)
- die angegebenen Pfade im Originalpaket stimmen nicht mit Ihrer lokalen Verzeichnisstruktur überein
- der Binärcode passt nicht zu Ihrer genutzten Plattform und Architektur
- das Format der Konfiguration sorgt für Ärger, bspw. die verfügbaren Optionen und Schalter
- es bestehen Konflikte mit anderer, bereits installierter Software

Verwenden Sie daher möglichst die Version der Software, die auch zu Ihrer Distribution und Architektur passt. Diese lässt sich i.d.R. am einfachsten in Ihren Softwarebestand integrieren. Weitere Informationen dazu finden auf der Projektwebseite von `alien` [alien].

22.2.2 Pakete umwandeln

22.2.2.1 Voraussetzungen

Damit Ihnen das Umwandeln von bestehenden Paketen in das `deb`-Format gelingt, müssen ein paar Voraussetzungen erfüllt sein. Für `alien` benötigen Sie:

- Perl [Debian-Paket-perl] — weil `alien` ein Perl-Skript ist
- die Werkzeuge `rpm` bzw. `yum` aus den gleichnamigen Debianpaketen für die Konvertierung von `rpm`-Paketen nach `deb` ([Debian-Paket-rpm] und [Debian-Paket-yum])
- die Werkzeuge `dpkg`, `dpkg-dev` und `debhelper` ([Debian-Paket-dpkg], [Debian-Paket-dpkg-dev], [Debian-Paket-debhelper]) zur Erzeugung von Debianpaketen
- der GNU-C-Compiler `gcc` und `make` ([Debian-Paket-gcc] und [Debian-Paket-make]), sofern Software aus dem Quellcode zu übersetzen ist
- das Paket *lsb* [Debian-Paket-lsb] zur Linux-Standardisierung gemäß der Linux Standard Base (LSB)

Die beiden Pakete `perl` und `dpkg` gehören zu den essentiellen Paketen und sind auf ihrer Debian-Installation vorhanden. Die anderen genannten Pakete könnten noch fehlen. Falls letzteres zutrifft, installieren Sie diese über die Paketverwaltung nach (siehe „Pakete installieren“ in Abschnitt 8.36).

22.2.2.2 Durchführung

- Umwandlung eines rpm in ein deb mittels expliziter Angabe des Paketformats über den Schalter `-d` (Langform `--to-deb`)

```
alien --to-deb paket.rpm
```

- `--to-deb` ist der Standardfall, d.h. kann man im Aufruf weglassen
- Alternative Formate:
  - rpm: `-r` (Langform `--to-rpm`)

  - tgz: `-t` (Langform `--to-tgz`)
  - slp: `--to-slp` (keine Kurzform)
  - pkg: (Open)Solaris-Pakete mittels `-p` (Langform `--to-pkg`)

Am Ende überprüfen Sie das Paket, um sicherzugehen, ob es auch tatsächlich zu ihrer bestehenden Installation passt.

- wie macht man diese Überprüfung?

22.2.2.3 Besonderheiten bei der Umwandlung

`deb`-basierte Systeme haben ihre Eigenheiten. Die folgenden Schalter von `alien` helfen Ihnen bei der Paketumwandlung:

- `--patch=Datei`, `--anypatch` und `--nopatch` zum automatischen Anpassen von Startup-Skripten und Pfaden gemäß dem File Hierarchy Standard (FHS)
- `-c` (Langform `--scripts`) um bestehende Pre- und Post-Install- sowie Remove-Skripte zu erhalten
- `-k` (Langform `--keep-version`), um die Versionsnummer beizubehalten — normalerweise zählt alien diese um eins hoch
- `-g` (Langform `--generate`) und `--veryverbose` um die Fehlersuche zu erweitern
- `-g` (Langform `--generate`), um das Paket vor der Umwandlung zu bearbeiten
  - erzeugt ein Verzeichnis mit dem Paketinhalt
  - ermöglich Ihnen die Ergänzung und Korrektur des Paketinhalts

22.2.3 Pakete umwandeln und einspielen

- Paket von rpm nach deb umwandeln und gleich einspielen (Kurzform)

```
alien -i paket.rpm
```

22.2.4 Umgewandelte Pakete einspielen

Haben Sie das Paket erfolgreich in das `deb`-Format umgewandelt, spielen Sie dieses mittels `dpkg -i` *paketname.deb* ein. APT und `aptitude` bekommen von der Aktion erstmal nichts mit, stören sich aber nicht daran, dass das Paket eingespielt ist.

Dabei können mehrere Ergebnisse eintreten — alles geht glatt und das Paket funktioniert, alles geht glatt und das Paket funktioniert nicht, oder das Einspielen geht schief. Da bleibt nur manuelle Nacharbeitung:

- was ist da zu tun?
- woran kann das hängen? (Ursachenforschung)

22.2.5 `deb`-Pakete in `rpm`-Strukturen

Auch für Linux-Distributionen, die auf dem `rpm`-Paketformat aufsetzen, können Sie `deb`-Pakete einspielen und nutzen. Möglich machen das die beiden Projekte *apt4rpm* [apt4rpm] bzw. *apt-rpm* [apt-rpm]. Während ersteres im Original von Conectiva stammt, steuerte RedHat das zweitgenannte bei.

Releases und entsprechende Pakete beziehen Sie von der jeweiligen Webseite des Projekts. Leider gibt es seit längerem keine neueren Veröffentlichungen mehr — *apt4rpm* schweigt seit 2005, *apt-rpm* seit 2008. 2012 verkündete *apt4rpm*, dass es die weitere Entwicklung einstellt.

- wie gut ist das dann überhaupt?
- wann braucht man das? wann kann das nützlich sein?
- wer kümmert sich darum?

Tipp
Wie Sie die beiden Werkzeuge verwenden können, entnehmen Sie bitte der Dokumentation der Werkzeuge.

Tipp
Für Arch Linux existiert *pacapt* [Arch-Linux-pacapt], welches die unterschiedlichen Paketmanager auf den einzelnen Plattformen anspricht. Derzeit verarbeitet es neben `dpkg` und `apt-get` auch `pacman` (Arch Linux, ArchBang), `homebrew` (Mac OS X), `yum`/`rpm` (RedHat, CentOS, Fedora) und `portage` (Gentoo).

Kapitel 23

Umgang mit LTS

Wie wir bereits in *Bedeutung der verschiedenen Entwicklungsstände* (siehe Abschnitt 2.10.1) beleuchtet haben, ist LTS eine Abkürzung und steht für *long-term support* — auf deutsch *Langzeitunterstützung*. Damit pflegt das Debianprojekt ältere Veröffentlichungen über bis zu fünf Jahre nach dem Ende des Releasezyklus [Plura-lts].

Betreiben Sie eine solche, ältere Debianinstallation, erreichen Sie irgendwann einen Punkt, an dem keine weiteren Aktualisierungen der verwendeten Softwarepakete mehr möglich sind. Versuchen Sie diesen Schritt trotzdem, meldet sich der Paketmirror mit dem Fehler „Release file expired“ bei Ihnen zurück. Zu diesem Zeitpunkt ist die LTS-Version „abgelaufen“, d.h. die weitere Pflege der Versionen aus dieser Veröffentlichung durch die Paketmaintainer ist eingestellt.

Können Sie die bislang verwendete Veröffentlichung nicht wechseln, stellt sich die Frage, ob — und wenn ja wie — Sie die bestehende Installation trotzdem noch weiter betreiben können. APT liefert dazu den Schalter `Acquire::Check-Valid-Until`, mit dem Sie die Überprüfung auf das Ende der Langzeitunterstützung überspringen und bei Bedarf auch vollständig abschalten können.

23.1 Kurzzeitiges Abschalten

Um die Überprüfung auf Gültigkeit des Release File einmalig zu ignorieren, rufen Sie `apt-get` mit dem Schalter `-o Acquire::Check-Valid-Until=false` auf. In Folge aktualisiert `apt-get` ihre Paketdatenbank anstandslos.

```
# apt-get -o Acquire::Check-Valid-Until=false update
```

23.2 Dauerhaftes Abschalten

Brauchen Sie das Abschalten häufiger — bspw. bis zum geplanten, tatsächlichen Versionswechsel — legen Sie am besten eine Datei im Verzeichnis `/etc/apt/apt.conf.d/` mit dem entsprechenden Schalter an. Mit dem nachfolgenden Aufruf erzeugen Sie eine Datei `/etc/apt/apt.conf.d/10no-check-valid-until` und setzen den Schalter dauerhaft [Stackexchange-LTS]:

```
# echo "Acquire::Check-Valid-Until=False;" > /etc/apt/apt.conf.d/10no-check-valid-until
```

Kapitel 24

Webbasierte Installation von Paketen mit `apturl`

In diesem Abschnitt beleuchten wir die Installation von `deb`-Paketen über das `apturl`-Protokoll (Langfassung: APT Protocol). Dahinter verbirgt sich eine Erweiterung für die Webbrowser Firefox/Iceweasel, Chromium, Konqueror, Opera und Epiphany (ab Gnome 3.6 umbenannt in Web). Voraussetzung dafür ist das Paket *apturl* [Ubuntu-apturl] von Michael Vogt [Vogt-apturl].

Das Paket ist bislang nicht in Debian enthalten, aber für Ubuntu als PPA verfügbar. Voraussetzung dafür ist mindestens Ubuntu 7 *Feisty Fawn* (veröffentlicht im Frühjahr 2007), in den neueren Veröffentlichungen gehört es bisher zur Basisinstallation.

24.1 Sinn und Zweck

Die Idee hinter *apt-url* ist die Installation von Paketen aus den eingetragenen Repositories (bzw. Channels bei Ubuntu) über die Adresszeile Ihres Webbrowsers. Für Installationen auf Servern und Desktops ist das nicht unbedingt der geeignete Weg, jedoch Tablet Computer, die Sie im wesentlichen via Webbrowser benutzen. Mitunter ist das die einzige Möglichkeit, auf dem Gerät eine Softwareerweiterung oder -aktualisierung vorzunehmen.

Um das Paket ‚cowsay' zu installieren, übertragen Sie diese Angabe in die Adresszeile des Webbrowsers und surfen die darüber genannte URL an:

```
apt://cowsay
```

apt-url akzeptiert die Angabe mehrerer Pakete auf einmal. Die einzelnen Paketnamen trennen Sie jeweils durch Komma voneinander. Leerzeichen sind dabei in der Paketliste nicht zulässig. Für die beiden Pakete `cowsay` und `xmms` sieht das wie folgt aus:

```
apt://cowsay,xmms
```

Außer den bereits o.g. Tablet Computern sehen wir noch weitere Einsatzbereiche:

- Sie benötigen ein bestimmtes Paket, aber die lokale Netzwerk-Infrastruktur, in der Sie sich befinden, gestattet nur den Zugriff über Port 80 (`http`) und 443 (`https`). Dieser Zustand trifft insbesondere auf freie WLANs bzw. WLANs in Hotels, öffentlichen Einrichtungen und Schulungsräumen zu, bei denen die Portfreigaben im Netzwerk recht restriktiv ausgelegt sind.
- Die Installation über die „üblichen Wege" geht nicht, aber `http` ist nutzbar.
- Ihr regulärer Paketmirror ist nicht erreichbar und Sie spielen das Paket aus einer lokalen Quelle als reguläre Datei ein.
- Sie nutzen einen lokalen, eigenen Paketmirror (siehe Paketverwaltung beschleunigen in Kapitel 25).

24.2 Risiken und Bedenken

Obwohl die Bereitstellung des Dienstes die Installation von Paketen über den Webbrowser sehr leicht macht, haben wir Bedenken im Umgang damit. Die nachfolgend genannten Punkte sind kaum durchführbar und bergen daher Risiken für die Integrität Ihres Systems:

- Überprüfung der Paketabhängigkeiten
- Überprüfung auf Vertrauenswürdigkeit der Paketquelle (siehe Abschnitt 3.12)
- Überprüfung auf Echtheit und Fehlerfreiheit des Pakets (siehe Abschnitt 8.35)

Bitte beachten Sie, dass alle Benutzer und jedes Programm oben genannte URL ansurfen und darüber eine Installation auslösen kann. Administrative Rechte sind zur Paketverwaltung in diesem Fall nicht mehr erforderlich und legen den Grundstein dafür, dass Aktivitäten außerhalb Ihres Sichtfeldes und ohne Ihr Wissen als Systemverantwortlicher passieren können.

24.3 apturl in der Praxis

- Todo:
  - Einsatzszenario
  - Empfehlungen zur Nutzung

Kapitel 25

Paketverwaltung beschleunigen

25.1 Hintergrund

- warum will man das:
  - Zeitersparnis
  - Menge der belegten Ressourcen möglichst verringern
  - Rechner wieder für die Aufgabe bereit haben, für die er gedacht ist
  - viele regelmäßige Updates ermöglichen
  - viele Rechner auf dem gleichen Stand halten und dabei den Aufwand möglichst minimieren
- bisher üblich:
  - jeder Rechner bezieht seine Pakete direkt vom Paketmirror und cacht diese selbst (lokal im Paketcache)
  - bei jedem Aufruf erfolgt eine separate Kommunikation mit dem Paketmirror
  - verfügbare Bandbreite der Internetanbindung/Leitung wird ausgelastet
    - Bezug der Pakete dauert länger
    - andere Transaktionen dauert auch länger (werden ausgebremst, sofern diese nicht mit höherer Priorität versehen sind)
  - Infrastruktur (Hardware) speichert Daten- und Netzwerkpakete zwischen
- viele Rechner hinter einem einzelnen Netzwerkzugang
  - werden alle aktualisiert, werden jedes Mal die gleichen Pakete erneut vom Paketmirror bezogen
  - bspw. automatisierte Aktualisierung (siehe Kapitel 31)

25.2 Möglichkeiten zur Beschleunigung

- Internetzugang durch dickere Leitung ersetzen
- Hardware (Router) durch leistungsstärkere Hardware ersetzen
- Aktualisierung zu den Zeiten vornehmen, wo es die wenigsten (realen) Benutzer stört oder beeinträchtigt
- eatmydata für (virtuelle) Wegwerfmaschinen
- `/var/cache/apt/` als tmpfs mounten, siehe Kapitel 27
- debdelta, siehe Kapitel 38

- PDiffs (gibt's nur bei Testing, Experimental und Sid), siehe auch Kapitel 38
- lokalen Zwischenspeicher für Pakete einrichten:
  - Proxy, der bereits herunterladene Pakete puffert (siehe Kapitel 26)
  - eigener Mirror (siehe Kapitel 28)
- weitere Softwarepakete zur Beschleunigung
  - *apt-fast* [apt-fast] und [Debian-Paket-apt-fast]
    - Shell-Wrapper um APT und `aptitude`
    - Download der Pakete über mehrere, parallele Verbindungen
    - benötigt den Downloadhelfer *aria2* [Debian-Paket-aria2] oder *axel*
    - weder in Debian noch in Ubuntu offiziell drin, nur per Launchpad-PPA
    - `apt-fast` akzeptiert alle Aufrufparameter von APT und `aptitude` und reicht diese einfach durch
    - Aufruf ändert sich in:

Aufruf der Paketinstallation mittels apt-fast

```
# apt-fast install cssed
...
#
```

25.3 Empfehlungen zum Umgang im Alltag

Die enorme Vielfalt des Debian-Paketbestands macht neugierig und ermutigt sicher nicht nur uns als Autoren, Software mit interessant klingenden Namen und Möglichkeiten auszuprobieren und damit zu experimentieren. Wir empfehlen Ihnen, trotz allem Enthusiasmus dabei die nachfolgenden Aspekte nicht zu vergessen:

- Mit der Zeit wächst die Menge installierter Softwarepakete, was a) zu einem gut gefüllten Speichermedium, b) zu potentiell mehr unbenutzt laufenden Programmen und c) zu einem Dateisystem führt, welches Stück für Stück an seine Leistungsgrenzen bzgl. der Einträge (Inodes) gelangt. Je stärker das Speichermedium gefüllt ist, umso größer wird die Zugriffszeit auf die Daten. Die Erfahrungswerte schwanken zwar, aber ein Füllstand von über 80% bedarf zumindest zunehmend stärkerer Beobachtung. Behalten Sie daher neben den Verzeichnissen für Binär- und Konfigurationsdateien auch den Paketcache im Auge (siehe dazu Kapitel 7).
- Halten Sie den Paketbestand auf Ihren Systemen möglichst klein und überschaubar. Damit verringert sich der Umfang der Pakete und Daten, die von Ihnen zu aktualisieren sind.
- Misten Sie den Softwarebestand regelmäßig aus. Lassen Sie daher nur die Softwarepakete installiert, die Sie auch tatsächlich benötigen. Dies setzt allerdings voraus, dass Sie wissen, was Sie oder die von Ihnen betreuten Benutzer tatsächlich verwenden. Als Admin sollten Sie das aber wissen bzw. können das sicher herausfinden. Vergessen Sie beim Aufräumen nicht die Konfigurationsdateien der Pakete (siehe Abschnitt 8.41). Achten Sie dabei insbesondere auf Pakete, die automatisiert Daemons starten. Dies erhöht die benötigte Rechenleistung und im Endeffekt auch die Stromrechnung.
- Setzen Sie die von Ihnen betreuten Systeme möglichst identisch auf. Das betrifft insbesondere die von Ihnen ausgewählte Veröffentlichung, den Paketbestand und die verwendeten Versionen. Es verringert damit die Vielfalt und Probleme, die überhaupt auftreten können (siehe dazu auch Kapitel 30).
- Aktualisieren Sie Ihre Software regelmäßig. Neben der sicherheitsrelevanten Aktualität des fehlerbereinigten Softwarebestands sorgt es für einen weiteren Zeitvorteil: viele kleine Änderungen sind meist einfacher durchzuführen als eine große mit vielen Korrekturen.
- Räumen Sie auch den Paketcache auf. Das kann sowohl automatisiert, als auch manuell geschehen (siehe dazu Abschnitt 7.3).

Kapitel 26

Einen APT-Cache einrichten

26.1 Begriff

Im Allgemeinen bezeichnet ein *cache* einen Datenpuffer oder Zwischenspeicher, der bereits angefragte Daten weiter vorhält und auf den im Bedarfsfall zurückgegriffen wird, wenn die Daten angefordert werden. Im vorliegenden Fall handelt es sich um eine Sonderform — einen lokalen Zwischenspeicher häufig benötigter Softwarepakete. Diese werden gepuffert und erlauben damit im Vergleich zum Bezug von Paketmirror aus dem Internet eine wesentlich geringere Zugriffszeit, da die Transportwege kürzer sind und weniger Zwischenstationen miteinbezogen werden müssen.

Das macht sich insbesondere dann deutlich bemerkbar, wenn mehrere Rechner diesen Datenpuffer verwenden und daraus die gleichen Softwarepakete beziehen. Liegt das angefragte Softwarepaket bereits im Cache vor, wird es von dort entnommen. Falls dieses jedoch noch nicht vorhanden sein sollte, wird es über das Internet vom Paketmirror bezogen und dem Cache hinzugefügt. Die nachfolgend Anfragenden erhalten dann das Softwarepaket deutlich schneller, da es dann bereits im Cache vorliegt und daraus bezogen werden kann.

Der APT-Cache funktioniert ähnlich wie der Puffer aus Kapitel 7, ist allerdings nicht rechnerspezifisch, sondern eher als Vorstufe für ein Netzwerk mit mehreren Clients gedacht und liegt meist auf einer eigenen Hardware (oder als Virtuelle Maschine) vor. Besonders zum Tragen kommt dieser Puffer bei einer von der Bandbreite her eher dünnen und stark schwankenden oder nur zeitweise verfügbaren Netzwerkanbindung. Erfolgt die Abrechnung Ihrer Internetanbindung entsprechend des transportierten Datenvolumens (sogenannte *trafficbasierte Abrechnung*), ermöglicht der APT-Cache eine Verringerung der Last nach außen und spart Ihnen somit bares Geld. Zudem wird die Bezugszeit von Softwarepaketen deutlich kleiner, was Ihre Produktivität steigern kann. Da auch ein externer Paketmirror ausfallen kann, lässt sich mit einem Paketcache die lokale Ausfallrate verringern.

Schematische Darstellung mit Paketcache in der Netzwerkinfrastruktur

```
* ToDo: Bild
```

Vor der Einrichtung ist zwischen einem Paketcache und dem eigenem Paketmirror abzuwägen (siehe Kapitel 28). Während letzterer stets alle Softwarepakete der Veröffentlichung vorhält — also auch die, die Sie nicht benötigen — landen im Paketcache hingegen nur die Pakete, die bisher tatsächlich angefragt wurden.

Abzuschätzen ist noch der Betreuungsaufwand, der bei beiden Varianten einzuplanen ist.

- ToDo: Betreuungsaufwand abschätzen

26.2 approx

- *approx* [Debian-Paket-approx]
- aus der Selbstbeschreibung:
  - HTTP-Proxyserver für debian-artige Paketarchive

  - Paketdepots müssen nur in den Konfigurationsdateien von *approx* geändert werden, und nicht in der `/etc/apt/sources.list`-Datei jedes Clients

- Approx kann ...
  - ... vorher verfügbare Konzepte wie *apt-proxy* ersetzen, ohne die `/etc/apt/sources.list`-Dateien der Clients zu ändern,
  - ... oder als Alternative zu *apt-cacher* eingesetzt werden (siehe Abschnitt 26.3).

26.3 apt-cacher

- Paket *apt-cacher* [Debian-Paket-apt-cacher]
- Beispiel:
  - *apt-mirror* und *apt-cacher*: [apt-mirror-ubuntu2]
- was *apt-cacher* tut:
  - klemmt netztechnisch zwischen dem (öffentlichen) Spiegelserver und den anfragenden Clients (bspw. auch einem Paketmirror)
  - interessant für kleine und größere Netzwerke, die (identisch) aufgesetzt wurden
  - `apt-cacher` ermöglicht einen Paketcache, d.h. es merkt sich, welche Pakete bereits angefragt wurden und liefert diese an den Anfragenden aus dem Cache, sofern sich das Paket bereits im Cache befindet, ansonsten holt es das Paket zuvor vom Paketmirror
- empfohlene Schritte, um das zu nutzen:
  - Installation der Pakete *apt-cacher*, idealerweise in Kombination mit einem APT-Proxy wie *apt-proxy* oder *approx* Abschnitt 26.2
  - Service aktivieren durch Anpassung der Datei `/etc/default/apt-cacher` und Änderung des Wertes für `AUTOSTART`: "AUTOSTART=0" in "AUTOSTART=1"
  - `apt-cacher` neu starten mittels `/etc/init.d/apt-cacher restart`
  - Clients konfigurieren, d.h. Bezugsadresse für Pakete ändern
    - neuen Service erzeugen und dazu die Datei `/etc/apt/apt.conf.d/90-apt-proxy.conf` anlegen
    - die Datei um den Eintrag ergänzen: `Acquire::http::Proxy "http://repository-cache:3142";`
    - `repository-cache` bezeichnet den Rechnernamen, der den Paketcache bereitstellt

26.4 apt-cacher-ng

- Programmpaket: *apt-cacher-ng* [Debian-Paket-apt-cacher-ng]
- Projektwebseite [apt-cacher-ng-Projektseite]
- Selbstbeschreibung:
  - Proxyserver zum Zwischenspeichern von Softwaredepots.
  - Proxy, um Pakete von Softwaredepots im Debian-Stil (oder möglicherweise von anderen Typen) herunterzuladen.
  - Downloads gehen durch den Proxy
  - Apt-Cacher NG speichert eine Kopie aller Nutzdaten, die über ihn laufen
  - Apt-Cacher NG wurde von Grund auf neu als Ersatz für `apt-cacher` entwickelt (siehe Abschnitt 26.3)
  - Fokus:
    - maximaler Durchsatz mit geringen Systemanforderungen
    - Ersatz für `apt-proxy` und `approx` (Abschnitt 26.2) verwendet werden, ohne die Datei `/etc/apt/sources.list` der Clients zu ändern.

26.5 debtorrent

- Debianpaket *debtorrent* [Debian-Paket-debtorrent]
- Projektseite [debtorrent-Projektseite]
- Idee:
  - Bezug von Debian-Paketen über das BitTorrent-Protokoll
  - gleichmäßiges und effektives Übertragen großer Datenmengen
  - Nutzung des Peer-to-Peer-Ansatzes
  - gemeinsames Ausnutzen der verfügbaren Bandbreite der Benutzer
- Paket, welches thematisch dazugehört:
  - *apt-transport-debtorrent* [Debian-Paket-apt-transport-debtorrent]

Kapitel 27

Cache-Verzeichnis auf separater Partition

Es gibt die unterschiedlichsten Gründe, entweder das gesamte Verzeichnis `/var` oder nur den Paketcache unter `/var/cache/apt/archives/` von den restlichen Partitionen in Ihrem Linuxsystem zu trennen. Die darin abgespeicherten variablen Daten sind in Bezug auf deren Größe und Menge (d.h. die Anzahl der Inodes für die Dateien) nicht vorhersehbar und schwanken. Aktualisieren Sie Software auf Ihrem Linuxsystem, möchten Sie verhindern, dass das Herunterladen der neuen Pakete Ihren Rechner durch ein vollbelegtes Dateisystem in der Benutzung ausbremst.

Auf Computern, bei denen das Betriebssystem entweder auf einer Solid State Disk (SSD), einer CompactFlash- oder Secure Digital Memory Card (kurz CF- bzw. SD-Karte) als einzigem Medium bereitsteht, möchten Sie die Anzahl der Schreibzugriffe möglichst reduzieren. Je seltener diese stattfinden, umso länger bleibt Ihnen der malträtierte Datenträger erhalten. Entsprechend lohnt sich hier ein Auslagern in einen externen Bereich. Das kann beispielsweise ein Share eines NFS-Servers sein, aber auch eine nur im Arbeitsspeicher existierende `tmpfs`-Partition[1]. Während für erstgenanntes ein schnelles lokales Netzwerk sowie die passende Infrastruktur Voraussetzung ist, zählt für die `tmpfs`-Partition im wesentlichen der verfügbare RAM.

Als erstes stellen wir Ihnen vor, wie Sie aus dem Verzeichnis `/var/cache/apt/archives/` eine RAM-basierte `tmpfs`-Partition erstellen. Danach besprechen wir, wie Sie einen Paketcache als separate Partition oder lediglich als Unterverzeichnis auf einer anderen Partition einrichten.

27.1 Paketarchiv als `tmpfs`-Partition

In **Schritt eins** überdenken Sie dazu die Größe der Partition. Wägen Sie dabei zwischen dem potentiellem und dem maximal zu erwartendem RAM-Verbrauch ab. Dabei helfen Ihnen diese Fragen:

- Wieviel RAM hat mein Rechner?
- Wieviel RAM kann ich im Zweifelsfall entbehren?
- Welche Größe haben die Pakete, die ich herunterladen will?

Beispiel: Bei einigen installierten Spielen, dem Officeprogramm LibreOffice, dem Webbrowser Chromium, dem Netzwerkanalysetool Wireshark und dem Texteditor GNU Emacs kann ein Satz Paketaktualisierungen schnell mal ein knappes Gigabyte ausmachen. Axel hat sich daher bei seinem Laptop mit 8 GB RAM für maximal 2 GB Platz für heruntergeladene Pakete entschieden.

Schritt zwei betrifft die Integration in die Verzeichnishierarchie. Tragen Sie dazu den neuen Einhängepunkt (engl. *mount point*) am Ende der Datei `/etc/fstab` ein. Bei Axels Laptop sieht dies wie folgt aus:

/etc/fstab-Eintrag für /var/cache/apt/archives/

[1] `tmpfs` (englisch für *temporary file system*) ist ein Dateisystem, das in vielen unix-artigen Betriebssystemen als verbesserter Ersatz für eine RAM-Disk eingesetzt wird. Im Gegensatz zur RAM-Disk, bei der realer Arbeitsspeicher verwendet wird, wird bei `tmpfs` virtueller Arbeitsspeicher statt der Festplatte als Speicher benutzt. (Quelle: Wikipedia)

```
# Gerät  Einhängepunkt            Dateisystemtyp  Optionen         fsck
none     /var/cache/apt/archives  tmpfs           size=2147483648  0  0
```

Alternativ können Sie anstatt einer festen Größe in Bytes (mit `size=<Größe in Bytes>`) auch einen prozentualen Wert angeben (`size=<Größe in Prozent der RAM-Größe>%`). Bei 8 MB RAM ist eine feste Größe von 2 MB mit der Angabe `size=25%` identisch.

Nur mit dem Eintrag selbst ist das Dateisystem aber noch nicht eingehängt. Um eventuell bereits heruntergeladene Pakete nicht zu verwerfen, verschieben Sie in **Schritt drei** als Benutzer `root` den aktuellen Inhalt des Verzeichnisses `/var/cache/apt/archives/` temporär woanders hin, hängen das `tmpfs`-Dateisystem aus dem RAM ein (es wird dabei automatisch erzeugt) und verschieben die vorher gesicherten Dateien an die alte Position im Dateibaum zurück.

Einhängen von /var/cache/apt/archives/ als tmpfs-Dateisystem

```
# mkdir /var/cache/apt/archives.temp
# mv /var/cache/apt/archives/* /var/cache/apt/archives.temp/
# mount /var/cache/apt/archives
# mv /var/cache/apt/archives.temp/* /var/cache/apt/archives/
#
```

Zum Testen rufen Sie am besten einmal das Kommando `apt-get update` auf. Wenn sich das Kommando nicht mit Fehlermeldungen gegen die aktuellen Einstellungen wehrt, haben Sie sehr wahrscheinlich alles richtig gemacht und die Einrichtung ist erfolgreich abgeschlossen.

Die Einrichtung als `tmpfs`-Dateisystem im RAM macht sich auch nach einem Neustart Ihres Linuxsystems bemerkbar. In diesem Fall verschwinden sämtliche heruntergeladenen Paketdateien wieder aus dem Cache. Ein Neustart hat somit den gleichen Effekt wie der Aufruf `apt-get clean` auf der Kommandozeile — nur: sie müssen seltener von Hand aufräumen.

27.2 Paketcache als separate Partition einrichten

Möchten Sie `/var/cache/apt/archives/` statt im RAM auf einer echten Partition oder einem NFS-Server haben, so ist der Ablauf zu obigem Beispiel nahezu identisch. Stattdessen ist zunächst noch die Partition und das Dateisystem anzulegen, welches als Paketcache dienen soll. Danach tragen Sie in der Datei `/etc/fstab` anstatt `none` den Gerätenamen oder NFS-Server plus Exportnamen. Anstatt von `tmpfs` setzen Sie den gewählten Dateisystemtyp (oder `nfs`) sowie die passenden Optionen (oder `default`). Da es an dieser Stelle viel zu viel Varianten gibt, um ein allgemeingültiges Beispiel zu geben, sei dies dem Leser als Übung überlassen.

27.3 Cache-Verzeichnis als Unterverzeichnis auf anderer Partition

Möchten Sie den Paketcache weder ins RAM verlagern noch dafür eine eigene Partition anlegen, sondern stattdessen einfach nur ein Unterverzeichnis auf einer anderen Partition benutzen, so bietet Ihnen APT diese Möglichkeit auch an. Dazu teilen Sie das APT mit und tragen Ihren Wunschpfad in der Konfigurationsdatei `/etc/apt/apt.conf` über die Option `Dir::Cache::archives` mit. Für den neuen Pfad `/scratch/paketcache/` zum Paketcache sieht der Eintrag beispielsweise wie folgt aus:

Beispiel für einen umgebogenen Paketcache in der Datei /etc/apt/apt.conf

```
Dir::Cache::archives "/scratch/paketcache/";
```

APT kann dieses Verzeichnis nicht selbst anlegen. Es verläßt sich daher darauf, dass dieses von Ihnen benannte Verzeichnis bereits existiert und nur Schreibrechte für den Benutzer `root` besitzt. Sollte das Verzeichnis noch nicht bestehen, legen Sie dieses wie folgt an:

Anlegen eines alternativen Verzeichnisses für den Paketcache

```
# mkdir /scratch/paketcache/
# chown root:root /scratch/paketcache/
# chmod 755 /scratch/paketcache/
#
```

Abschließend überprüfen Sie, ob Ihre Eintragung wie erhofft funktioniert. Dabei hilft Ihnen der kombinierte Aufruf von `apt-config` und `fgrep`. `apt-config` mit der Option `dump` liefert Ihnen die aktuelle Konfiguration von APT, aus der Sie mittels `fgrep` nach dem Eintrag `Dir::Cache` suchen.

Überprüfen der geänderten Dir::Cache-Einstellung für APT

```
$ apt-config dump | fgrep Dir::Cache
Dir::Cache "var/cache/apt/";
Dir::Cache::archives "/scratch/paketcache/";
Dir::Cache::srcpkgcache "srcpkgcache.bin";
Dir::Cache::pkgcache "pkgcache.bin";
$
```

Es ist nicht auszuschließen, dass das eine oder andere Programm den üblichen Pfad `/var/cache/apt/archives/` hart verdrahtet hat und gar nicht in der APT-Konfiguration nach einem geänderten Pfad schaut (was ein Bug wäre). Daher ist es empfehlenswert, zusätzlich einen entsprechenden Hinweis zu hinterlegen. Am einfachsten ist ein symbolischer Verweis (engl. kurz: *Symlink*) vom üblichen Pfad `/var/cache/apt/archives/` zum neuen Ort[2]. Auch hier gilt wieder: Am besten gleich nach dem Einrichten mit dem Kommando `apt-get update` testen und ggf. korrigieren.

Symlink zum neuen Paketcache anlegen

```
# ln -s /scratch/paketcache/ /var/cache/apt/archives/
#
```

Konfiguration von APT und apt-config

Wie Sie APT auf Ihre spezifischen Bedürfnisse anpassen, lesen Sie in Kapitel 10. Detaillierter gehen wir auf das oben genutzte Kommando `apt-config` unter Konfiguration von APT anzeigen in Abschnitt 10.2 ein.

---

[2] Im Normalfall sollte sogar der Symlink alleine auch ausreichen, damit APT einen alternativen Cache findet. APT folgt dann einfach auch dem Symlink.

Kapitel 28

Einen eigenen APT-Mirror aufsetzen

Wie bereits in Abschnitt 3.4 deutlich wurde, stellen die offiziellen Paketmirrors stets aktuelle Debian-Pakete für alle Interessierten zur Verfügung. Diese Spiegelserver werden weitestgehend individuell betreut.

Mit einem Paketmirror verfolgen Sie das gleiche Ziel wie mit einem Paketcache (siehe Einen APT-Cache einrichten in Kapitel 26). Darüber werden ebenfalls Debian-Softwarepakete in einer Art Puffer bereitgestellt. Ziele sind dabei einerseits die Unabhängigkeit zu externen Paketquellen und andererseits der Wunsch, die Bezugszeiten über das lokale Netzwerk für die Arbeitsgruppe oder Wirtschaftseinheit möglichst zu verringern.

Der Unterschied zwischen einem Paketcache und einem Paketmirror besteht darin, dass bei ersterem nur die bereits nachgefragten Pakete vorgehalten werden, während ein Paketmirror i.d.R. die gesamte Distribution inkl. ausgewählter Veröffentlichungen bereitstellt. Ein lokaler Paketmirror spiegelt den offiziellen Paketbestand und macht ihn für die Benutzer, Dienste und Geräte in einem lokalen Netzwerk verfügbar. Damit geht eine Entlastung der offiziellen Paketmirror und eine Verringerung des Netzwerkverkehrs zu diesem einher — die verfügbare Bandbreite der Außenanbindung kann anderweitig genutzt werden. Es ist an der Stelle jedoch dafür zu sorgen, dass der lokale Paketmirror stets aktuelle Pakete vorhält.

Nachfolgend stellen wir Ihnen mehrere Varianten anhand der Pakete *apt-mirror* (Abschnitt 28.1), *debmirror* (Abschnitt 28.2), *debpartial-mirror* (Abschnitt 28.3) und *reprepro* (Abschnitt 28.4) vor.

28.1 apt-mirror

- Paket *apt-mirror* [Debian-Paket-apt-mirror]
  - bei Ubuntu verfügbar im Zweig *universe*
- Projektseite [apt-mirror-Projektseite]
- geeignet zum Spiegeln der Debian-Distribution oder einer beliebigen APT-Quelle
- Funktionsumfang (aus der Paketbeschreibung):
  - Es verwendet eine gleichartige Konfiguration wie die `/etc/apt/sources.list` von APT
  - Es stimmt vollständig mit dem Paket-Pool überein.
  - Es kann beim Herunterladen mit mehreren Threads arbeiten.
  - Es unterstützt mehrere Architekturen gleichzeitig.
  - Es kann nicht benötigte Dateien automatisch entfernen.
  - Es funktioniert auch mit überlasteten Internetverbindungen gut.
  - Es erstellt nie einen inkonsistenten Spiegel, sogar während des Spiegelns.
  - Es funktioniert auf allen POSIX-konformen Systemen mit `perl` und `wget` [Debian-Paket-wget]

28.1.1 Wichtige Dateien aus dem Paket

- `/etc/apt/mirror.list`
- `/etc/cron.d/apt-mirror` oder `crontab`
- `apt-mirror`

28.1.2 Ablauf

- Paket *apt-mirror* beziehen und installieren
- `/etc/apt/mirror.list` anpassen

Beispielkonfiguration für /etc/apt/mirror.list

```
set nthreads     5
set _tilde 0

deb http://ftp.de.debian.org/debian/ wheezy main contrib non-free
deb http://security.debian.org/ wheezy/updates main contrib non-free

# backports
deb http://ftp.de.debian.org/debian wheezy-backports main contrib non-free

# cleaning section
clean http://ftp.de.debian.org/debian/
clean http://security.debian.org/
clean http://ftp.de.debian.org/debian
```

- `/etc/cron.d/apt-mirror` oder `crontab` anpassen

Eintrag für eine tägliche Aktualisierung des Paketmirrors

```
@daily /usr/bin/apt-mirror
```

- das Kommando `apt-mirror` ausführen
  - Pakete landen danach in `/var/spool/apt-mirror`
- Mirror via http verfügbar machen
  - Symlink zum Apache-Verzeichnis anlegen

```
# ln -s /var/spool/apt-mirror/mirror/ftp.de.debian.org/debian/ /var/www/html/debian
```

- `/etc/apt/sources.list` der Clienten entsprechend anpassen :)

28.1.3 Beispiel/HowTo

- nur *apt-mirror*: [apt-mirror-ubuntu]
- *apt-mirror* und *apt-cacher*: [apt-mirror-ubuntu2]

28.1.4 Hinweise

- nur die Bereiche und Architekturen auswählen, die Sie tatsächlich benötigen — alles andere braucht *sehr viel* Platz
  - Ubuntu: etwa 15G pro Architektur
- initialer Bezug kann sehr lang dauern
  - bei einem Folgebezug werden nur die Änderungen übertragen — daher geht das deutlich schneller

28.2 debmirror

- Paket: [Debian-Paket-debmirror]
- Beschreibung von *debmirror* unter http://debiananwenderhandbuch.de/debianmirror.html
- keine Konfigurationsdatei
  - stattdessen: alles als Parameter für den Aufruf über die Kommandozeile
- Parameter/Optionen (Auswahl):

  `-a`
  Architektur

  `-s`
  Section

  `-d`
  Distribution bzw. Veröffentlichung

  `-h`
  Host oder Server, von dem bezogen werden soll

  `--nosource`
  keine Source-Pakete

  `--progress`
  Verlauf anzeigen

  `-v`
  ausführliche Ausgabe (verbose)

  `--method`
  Methode, die zum Bezug verwendet werden soll

Beispielaufruf für debmirror in /home/ftp/debian

```
# debmirror -a i386 -s main -s contrib -s non-free \
 -h ftp.debian.org \
 -d wheezy
 /home/ftp/debian \
 --nosource \
 --progress
#
```

28.3 debpartial-mirror

- Paket: [Debian-Paket-debpartial-mirror]
- geeignet zum Spiegeln eines partiellen Debian-Mirrors
  - wenn man nur einen Ausschnitt des Paketbestands benötigt
  - wenn nur begrenzt Platz vorhanden ist
- aus der Paketbeschreibung:

„With it, you can mirror selected sections, priorities, packages, virtual packages, or even files and directories matching regular expressions. Packages may be drawn from any number of sources, with dependencies for each source resolved from whichever other sources you specify.“

28.4 reprepro

- Paket [Debian-Paket-reprepro]
- Optionen und Filter, um ein Repo zu erzeugen

Kapitel 29

Plattenplatz sparen mit der Paketverwaltung

- Hardlink über `/usr/share/doc/` laufen lassen [Beckert-Blog-Hardlinking-Duplicate-Files]
- `dpkg --path-exclude=...`
- Paket *localepurge*, momentan noch unter Kapitel 36
- Ungenutzte Bibliotheken, denen aber das „Automatisch installiert“-Flag fehlt [Beckert-Blog-Finding-Libraries]
- Auch `/var/cache/apt` auf tmpfs gehört hier erwähnt, momentan unter Kapitel 27
- `logrotate` (sollte eigentlich installiert sein, sorgt aber für nicht permanent wachsende Logfiles. Seine Konfiguration sollte angepasst werden, wenn man manuell weitere Logfiles in Anwendungen (VHosts im Apache z.B.) konfiguriert.
- Keinen Syslogd (Default-Paket ist *rsyslog*, dieses entfernen) verwenden und das Syslog nur in einen Ringbuffer laufen lassen.
  - Hat man systemd, kann man mit `journalctl` das Log im Ringbuffer anschauen.
  - Hat man sysvinit, so kann man das Paket *busybox-syslogd* installieren und mit `logread` den Inhalt des Ringbuffers anschauen.
  - Gehört eigentlich nicht zur Paketverwaltung… Kann man entsprechend zum Kürzen wieder rauskippen.

Kapitel 30

Automatisierte Installation

- ganze Rechnerverbünde (Schulungs- und Klassenräume) mit identischer Hardware
- Vorbereitung von Schulungsrechnern
- https://wiki.debian.org/AutomatedInstallation
- Historie
  - jumpstart (Sun OS) — http://en.wikipedia.org/wiki/Jumpstart_(Solaris)
  - Kickstart und Cobbler — http://www.cobblerd.org/
  - FAI
  - Preseeding — Beantworten der Fragen des Debian Installers im Vorfeld

30.1 FAI

- Eintrag im Debian Wiki [Debian-Wiki-FAI]

30.2 Kickstart

- http://cobbler.github.com/
- http://www.debian.itopstube.com/2011/11/automating-installation-with-kickstart.html
- https://fak3r.com/2011/08/18/howto-automate-debian-installs-with-preseed/

30.3 Erfahrungen aus der Praxis

ToDo

Kapitel 31

Automatisierte Aktualisierung

- Ziel:
  - eine ganze Menge von identisch aufgesetzten Rechnern aktualisieren, bspw. einen Schulungsraum oder Cluster

31.1 `apt-dater`

- Werkzeug/Paket: *apt-dater* [Debian-Paket-apt-dater]
- terminal-based remote package update manager
- Projektseite [apt-dater-Projektseite]
- Kurzbeschreibung: "apt-dater provides an ncurses frontend for managing package updates on a large number of remote hosts using SSH. It supports Debian-based managed hosts as well as rug (e.g. openSUSE) and yum (e.g. CentOS) based systems."
- benötigt SSH und Screen

Kapitel 32

Qualitätskontrolle

Ihre Debian-Installation besteht aus recht vielen Paketen, an denen etliche Entwickler beteiligt sind. Wie in Abschnitt 2.14 schon deutlich wurde, kümmert sich das *Debian Quality Assurance Team* (kurz *QA Team*) [DebianQA] darum, dass die Qualität der Debian-Pakete gewährleistet ist. Dazu gehört, dass auch alle vorab festgelegten Regeln eingehalten werden. Um das automatisiert zu prüfen, kommen dafür eine ganze Reihe von Programmen zum Einsatz.

Nachfolgend stellen wir Ihnen mehrere dieser Werkzeuge vor, mit denen Sie eine solche Inspektion selbst vornehmen und nachvollziehen können. Für noch nicht installierte, einzelne Pakete besprechen wir `lintian`, bereits installierte Pakete verifizieren wir stattdessen mittels `adequate`. Für die Recherche in den Fehlerberichten (engl. *bug reports*) zeigen wir Ihnen den Umgang mit `apt-listbugs`, `apt-listchanges`, `popbugs`, `rc-alert` und `how-can-i-help`. Im Rahmen der Betreuung älterer Installationen ist das Programm `debian-security-support` von großem Nutzen.

Möchten Sie den gesamten Installationsvorgang eines Pakets testen, steht Ihnen die *Package Installation, UPgrading And Removal Testing Suite (Piuparts)* [Piuparts] aus dem gleichnamigen Debianpaket [Debian-Paket-piuparts] zur Seite. Die drei Werkzeuge `piuparts`, `lintian` und `adequate` ergänzen einander und helfen Ihnen insbesondere bei der aktiven Verifikation von Paketen aus dem eigenen Paketlabor, bevor Sie diese in die freie Wildbahn entlassen.

32.1 Nicht installierte Pakete mit `lintian` prüfen

32.1.1 `lintian` verstehen

Das Werkzeug `lintian` [Lintian] steht in dem gleichnamigen Paket [Debian-Paket-lintian] bereit. Der Name leitet sich von engl. *lint* für Fussel und dem *ian* aus *Debian* ab. Es analysiert die verschiedenen Bestandteile eines einzelnen Debianpakets hinsichtlich typischer Fehler und insbesondere auch bzgl. häufig vorkommender Verstöße gegen die Debian-Richtlinien (*Debian Policy Violations*).

Als Systemadministrator hilft Ihnen `lintian` primär dabei, sowohl eigene als auch die Pakete von Drittparteien aus nicht-Debian-eigenen Paketquellen auf grundlegende Probleme hin zu testen. Deswegen gehen wir an dieser Stelle vor allem auf die Nutzung von `lintian` über die Kommandozeile ein.

Bei Debian wird `lintian` primär an drei verschiedenen Stellen genutzt:

- Testen frisch gebauter Pakete durch den Entwickler,
- allgemeine Qualitätskontrolle des Paketbestandes (siehe Abbildung 32.1) und
- automatisierte Ablehnung von frisch hochgeladenen Paketen bei groben Fehlern[1].

Dazu führt `lintian` eine ganze Reihe vorbereiteter Tests mit dem Paket durch. Das Ergebnis umfasst Fehlermeldungen mit unterschiedlichem Schweregrad, deren Kategorien wir für Sie in Tabelle 32.1 zusammengestellt haben.

[1] Von `lintian` bemerkte, besonders schwere Fehler sollten bei offiziellen Paketen gar nicht auftauchen, da diese damit sozusagen bereits beim Aufnahmetest durchfallen.

Tabelle 32.1: Klassifikation der `lintian`-Fehlermeldungen

Parameter	Beschreibung
`E`	Fehler (*error*)
`W`	Warnungen (*warning*)
`I`	Informationelle Hinweise (*informational tags*)
`P`	Pingelige Markierung (*pedantic tags*)
`O`	Überschriebene Markierungen (*overridden tags*)
`X`	Experimentelle, ggf. fehleranfällige Markierungen (*experimental tags*)
`N`	Kein Fehler, Bemerkung (*note*)

32.1.2 `lintian` verwenden

`lintian` arbeitet auf einzelnen, vorbereiteten Paketdateien, nicht jedoch auf bereits installierten Paketen. Für letzteres eignet sich hingegen das Paket *adequate*, welches wir in Abschnitt 32.2 genauer besprechen.

Das Programm verarbeitet sowohl Dateien für Quellpakete (`.dsc`), als auch für Binärpakete (`.deb`). Übergeben Sie `lintian` auch die während des Paketbaus erstellte Datei `.changes`, in welcher alle Dateien des jeweiligen Quell- und Binärpakets aufgelistet sind, validiert das Programm nacheinander beide Entwicklungsstufen in einem Rutsch.

Im nun folgenden Beispiel überprüft `lintian` das Paket *mp4h*. Als Schalter kommen zum Einsatz:

`-v` (Langform `--verbose`)
: für eine ausführlichere Ausgabe (*verbose*).

`--color auto`
: für eine farbige Kennzeichnung des Schweregrads des gefundenen Fehlers bei einer Ausgabe im Terminal. Zulässig sind ebenso die Werte `never` (keine Hervorhebung), `always` (stets mit Hervorhebung) und `html` (Hervorhebung bei der Ausgabe als Webseite).

`-I` (Langform `--display-info`)
: Auflistung der informationellen Markierungen (*info tags*). So bleiben auch Schreibfehler nicht unentdeckt.

`-E` (Langform `--display-experimental`)
: Auflistung der experimentellen Markierungen.

`--pedantic`
: Legt eine noch genauere Überprüfung des Pakets fest.

Weitere Schalter und Parameter sind in der Manpage zu `lintian` ausführlich erklärt.

Aufruf von lintian zum Finden typischer Probleme im Paket mp4h

```
$ lintian -v --color auto -I -E --pedantic mp4h_1.3.1-9_amd64.changes
N: Using profile pkg-perl/main.
N: Setting up lab in /tmp/temp-lintian-lab-1SvMHn5dUB ...
N: Unpacking packages in group mp4h/1.3.1-9
N: ----
N: Processing changes file mp4h (version 1.3.1-9, arch source amd64) ...
N: ----
N: Processing source package mp4h (version 1.3.1-9, arch source) ...
P: mp4h source: no-dep5-copyright
W: mp4h source: ancient-standards-version 3.9.4 (current is 3.9.6)
```

```
P: mp4h source: debian-watch-may-check-gpg-signature
N: ----
N: Processing binary package mp4h (version 1.3.1-9, arch amd64) ...
P: mp4h: no-homepage-field
W: mp4h: postinst-has-useless-call-to-ldconfig
W: mp4h: postrm-has-useless-call-to-ldconfig
$
```

Die Ausgabe auf der Kommandozeile und in Abbildung 32.1 sind aus mehreren Gründen nicht deckungsgleich. Es wurde zwar jeweils die gleiche Paketversion überprüft, aber dabei kamen unterschiedliche `lintian`-Versionen zum Einsatz. Auf der Projektseite wird meist die neueste `lintian`-Variante aus Debian *unstable* oder *testing* genutzt, die sich z.B. auch anhand der durchgeführten Tests unterscheidet, welche in das Paket aus Debian *stable* eingeflossen sind.

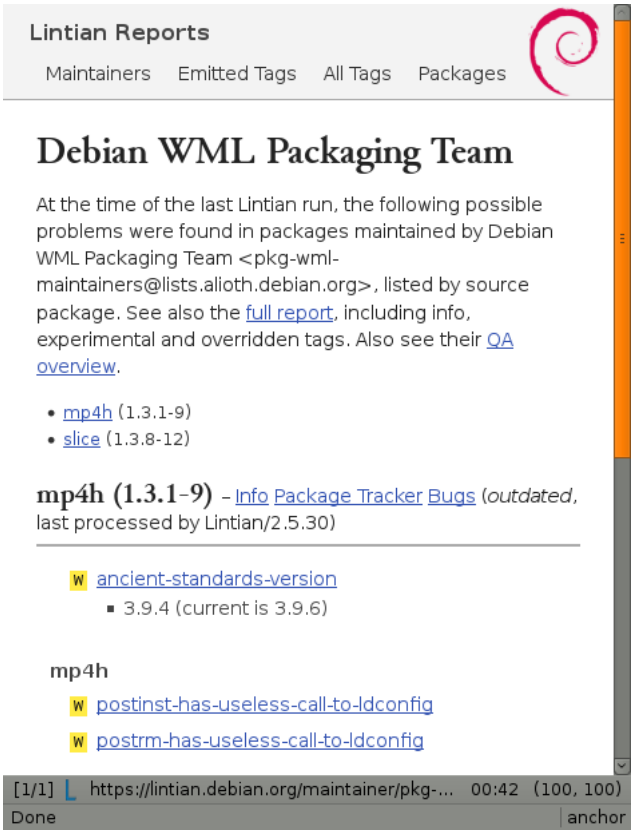

Abbildung 32.1: Ausgabe der Paketvalidierung mittels `lintian` zum Paket *mp4h*

Hilfreich ist auch die Option `-i` (Langform `--info`). Damit erhalten Sie bei jedem ersten Vorkommen einer Markierung noch zusätzliche Erklärungen und ersehen daraus, was die jeweilige Markierung bedeutet. Den gleichen Effekt erhalten Sie, wenn Sie die Ausgabe von `lintian` (ohne den Schalter `-v`) über eine Pipe an das Kommando `lintian-info` weiterleiten. `lintian-info` ist ebenso Bestandteil des Pakets *lintian*. Nachfolgend sehen Sie einen Ausschnitt zur Ausgabe dieses Programmaufrufs, bei dem das Paket *mp4h_1.3.1-9_amd64.deb* überprüft wird.

Erklärung zu den von lintian gefundenen Problemen im Binärpaket mp4h (Ausschnitt)

```
$ lintian -I -E --pedantic mp4h_1.3.1-9_amd64.deb | lintian-info
P: mp4h: no-homepage-field
N:
N:   This non-native package lacks a Homepage field. If the package has an
N:   upstream home page that contains useful information or resources for
N:   the end user, consider adding a Homepage control field to
N:   debian/control.
```

```
N:
N:   Refer to Debian Policy Manual section 5.6.23 (Homepage) for details.
N:
N:   Severity: pedantic, Certainty: possible
N:
N:   Check: fields, Type: binary, udeb, source
N:
W: mp4h: postinst-has-useless-call-to-ldconfig
N:
N:   The postinst script calls ldconfig even though no shared libraries are
N:   installed in a directory controlled by the dynamic library loader.
N:
N:   Note this may be triggered by a bug in debhelper, that causes it to
N:   auto-generate an ldconfig snippet for packages that does not need it.
N:
N:   Refer to Debian Policy Manual section 8.1.1 (ldconfig) and
N:   http://bugs.debian.org/204975 for details.
N:
N:   Severity: minor, Certainty: certain
N:
N:   Check: shared-libs, Type: binary, udeb
[...]
$
```

32.2 Bereits installierte Pakete mit `adequate` prüfen

Im Gegensatz zu `lintian` (siehe Abschnitt 32.1) validieren Sie mit `adequate` [Debian-Paket-adequate] die Pakete, die bereits auf ihrem System installiert sind. `adequate` steht Ihnen ab Debian 8 *Jessie* oder über die *Debian Backports* für Debian 7 *Wheezy* bereit.

Zur Paketanalyse versteht es die folgenden Schalter (Auswahl):

Paketname
: Überprüfung des von Ihnen angegebenen Debianpakets.

`--all`
: Überprüfung aller Pakete, die auf ihrem Debian-System derzeit installiert sind.

`--tags` ***Tag1 -Tag2***
: Beschränkung der Ausgabe auf die angegebenen Fehlerwerte zu *Tag1* und ohne *Tag2*. Als Tag sind bspw. *broken-symlink*, *missing-copyright-file* und *program-name-collision* zulässig.

Weitere Schalter und die vollständige Liste der Tags entnehmen Sie bitte der Manpage zum Programm.

Im ersten Beispiel sehen Sie das Ergebnis der Validierung des Pakets *pdfstudio*, welches aus einer nicht-Debian-eigenen Paketquelle stammt. In diesem Fall hat `adequate` entdeckt, dass die Informationen zum Copyright des Werkzeugs fehlen.

Überprüfung des Pakets pdfstudio mit Fehlermeldung

```
$ adequate pdfstudio
pdfstudio: missing-copyright-file /usr/share/doc/pdfstudio/copyright
$
```

Die Validierung ihres gesamten Systems erfolgt mit Hilfe des Schalters `--all` und wird durchaus etwas mehr Zeit in Anspruch nehmen. Nachfolgend sehen Sie einen Ausschnitt des Ergebnisses für ein Desktopsystem auf der Basis von Debian 7 *Wheezy*, welches sich bereits über eine längere Zeit in Verwendung befindet.

Überprüfung des gesamten Paketbestands (Ausschnitt)

```
$ adequate --all
libc-bin: program-name-collision /usr/bin/rpcinfo /usr/sbin/rpcinfo
rpcbind: program-name-collision /usr/sbin/rpcinfo /usr/bin/rpcinfo
virtuoso-opensource-6.1-bin: incompatible-licenses /usr/bin/isql-vt OpenSSL (libssl.so ↩
    .1.0.0) + GPLv3+ (libreadline.so.6)
exifprobe: broken-symlink /usr/share/doc/exifprobe/changelog.gz -> ../CHANGES_2.0.gz
python-matplotlib-data: broken-symlink /usr/share/matplotlib/mpl-data/fonts/ttf/cmsy10.ttf  ↩
    -> ../../../../fonts/truetype/ttf-lyx/cmsy10.ttf
...
$
```

32.3 Bugreports anzeigen

32.3.1 Hintergrundwissen

Allgemein gesprochen, ist ein Bugreport ein **Fehlerbericht** zu einem aufgetretenen bzw. von Ihnen bemerkten Fehler einer Soft- oder Hardware. Bei Debian ist der Bericht und dessen Einreichung vom Ablauf und der Form her standardisiert, damit dieser entdeckte Fehler möglichst automatisiert über das 'Debian Bug Tracking System (Debian BTS) [Debian-Bug-Tracking-System] verarbeitet und von allen Benutzern nachverfolgt werden kann. Somit ordnen Sie einen Fehlerbericht leichter einem bestimmten Paket oder einer spezifischen Systemkonstellation zu.

Generelles Ziel ist dabei, die bereits bestehenden Softwarepakete zu verbessern und auch noch nicht als stabil gekennzeichnete Pakete auf Fehler hin zu überprüfen und zu bereinigen. Dazu zählen auch Verstöße gegen (Debian)Richtlinien und Qualitätsvorgaben. Vor der Bereitstellung (Veröffentlichung) eines Pakets wird dann automatisch geprüft, ob das Paket den Anforderungen (Richtlinien) entspricht.

Weiterhin zählt dazu auch eine Prüfung auf kritische Fehler vor der Installation eines Pakets. `apt-get` wertet dazu das Ergebnis von `apt-listbugs` (siehe Abschnitt 32.3.2) aus und warnt Sie, falls für das Paket kritische Fehler im Debian BTS hinterlegt sind. Die Entscheidung liegt dann bei Ihnen, ob Sie das Paket wirklich installieren möchten oder lieber nicht.

An der **Recherche nach Fehlern** darf sich jeder Debian-Benutzer beteiligen. Dafür benutzen Sie das Debian BTS, um darin einerseits nach bestehenden Softwarefehlern und deren Lösungen zu recherchieren und andererseits weitere Fehler zu berichten, die Ihnen aufgefallen sind. Für ersteres muss das ganze reproduzierbar sein und es darf sich nicht um einen Bedienfehler handeln.

Wie bereits oben benannt, finden Sie bereits bekannte und eingetragene Fehler in der Fehlerdatenbank des Debian BTS. Über das webbasierte System suchen Sie anhand des Paketnamens, der Veröffentlichung, des Schweregrads, der Betreffzeile des Fehlerberichts, der Fehlernummer, dem Namen des Einreichenden, dem Status der Bearbeitung des Fehlers oder dem Bearbeiter — also demjenigen, der sich um die Bereinigung des Fehlers kümmert. Auf der Kommandozeile stehen Ihnen die Werkzeuge `lintian` (siehe Abschnitt 32.1), `apt-listbugs` (siehe Abschnitt 32.3.2) und `popbugs` (siehe Abschnitt 32.3.4) zur Verfügung.

Finden Sie einen Fehler bezüglich eines Debianpakets, schreiben Sie am besten einen Fehlerbericht (*bug report*). Eine ausführliche Beschreibung dessen, auf welche Punkte Sie bei dem Fehlerbericht wertlegen sollten, finden Sie auf der Webseite des Debian BTS. Bei der Zusammenstellung des Fehlerberichts hilft Ihnen das Werkzeug `reportbug` aus dem gleichnamigen Debianpaket [Debian-Paket-reportbug].

32.3.2 Bugreports mit `apt-listbugs` lesen

Die generelle Idee zu dem Werkzeug `apt-listbugs` aus dem gleichnamigen Debianpaket [Debian-Paket-apt-listbugs] ist, Fehlerberichte aus dem Debian BTS abzurufen. Wie wir bereits zuvor in Abschnitt 32.3.1 angerissen haben, ist das Werkzeug in den Ablauf zur Aktualisierung und Installation eines Pakets mit APT integriert. Es prüft in diesem Zusammenhang automatisch mit, ob im Debian BTS Fehler für das betreffende Paket vorliegen und diese bereits repariert wurden.

Darüberhinaus können Sie das Werkzeug auch direkt über die Kommandozeile aufrufen und sich eine Liste der registrierten Fehler ausgeben lassen. `apt-listbugs` akzeptiert dafür die folgenden Schalter (Auswahl):

`-s` *Schweregrad* (Langform `--severity`)
Fehler je nach Schweregrad eingrenzen. Möglich sind die Werte `critical` (kritisch), `grave` (sehr schwerwiegend), `ser`

ious (schwerwiegend), important (wichtig), normal (normal), minor (weniger relevant), wishlist (Wunschliste) und all (alle Schweregrade). Mehrere Werte trennen Sie mittels Komma voneinander. Der Standardwert ist die Kombination der drei erstgenannten Werte critical,grave,serious.

-T *Schlüsselworte* (Langform --tags)
Filtere die Fehlermeldungen anhand des gegebenen Schlüsselworts. Mehrere Werte trennen Sie mittels Komma voneinander.

-S *Status* (Langform --stats)
Filtere und sortiere die Fehlerberichte anhand des aktuellen Status. Mögliche Statuswerte sind pending (offener Fehler), forwarded (der Fehlerbericht ist als weitergeleitet markiert), pending-fixed (der Fehlerbericht ist als *gelöst* markiert, aber noch ohne Bestätigung), fixed (der Fehlerbericht ist markiert als *gelöst*), absent (in der angefragten Veröffentlichung bzw. Architektur existiert der Fehler nicht) und done (für die angefragte Veröffentlichung bzw. Architektur ist der Fehler gelöst).

-B *Fehlernummer* (Langform --bugs)
Filtere die Fehlerberichte anhand der gegebenen Nummer des Fehlerberichts und zeige nur die betreffenden an. Mehrere Werte trennen Sie mittels Komma voneinander.

-D (Langform --show-downgrade)
Zeige nur die Fehlerberichte für Pakete an, für die ein Downgrade erfolgt ist (siehe auch Abschnitt 8.40).

-P *Priorität* (Langform --pin-priority)
Benutze die Pin-Priorität als Filterkriterium (siehe Kapitel 20 für weitere Informationen zur Pin-Priorität).

Als weiteren Wert zum Aufruf benötigt apt-listbugs noch ein Kommando. Zur Auswahl stehen apt, list und rss. Bei ersterem liest apt-listbugs von stdin, bei list erwartet es die Paketnamen als Argumente zum Aufruf und bei letzterem im RSS-Format. Den Abschluss des Aufrufs bildet der Paketname, an den Sie zudem eine spezifische Paketversion anfügen können. Als Trennzeichen fungiert hier der Schrägstrich, sodass bspw. die Spezifikation für das Paket *apt-listbugs* in der Version 0.1.5 apt-listbugs/0.1.5 lautet.

Berichtete Fehler zum Paket coreutils auflisten

```
$ apt-listbugs -s critical,grave,serious list coreutils
Laden der Fehlerberichte ... Erledigt
»Found/Fixed«-Informationen werden ausgewertet ... Erledigt
grave Fehler von coreutils (-> ) <ungelöst>
 #743955 - coreutils: corrupted files on heavily fragmented ext3 and ext4 partitions
Zusammenfassung:
 coreutils(1 Fehler)
$
```

32.3.3 Ergänzende Bugreports mit apt-listchanges herausfiltern

Während Ihnen apt-listbugs alle Bugreports anzeigt, vergleicht apt-listchanges aus dem gleichnamigen Paket [Debian-Paket-apt-listchanges] die Änderungen zwischen dem lokal installierten Paket und der neuen, verfügbaren Version auf dem Spiegelserver. Dazu wertet es die beiden Dateien NEWS.Debian und changelog.gz bzw. changelog.Debian.gz aus.

Die Ausgabe von apt-listchanges steuern Sie dabei über mehrere Schalter (Auswahl):

-a (Langform --all)
Ausgabe aller Änderungen des Pakets.

`-f` *Ausgabegerät* (Langform `--frontend`)
Legt fest, an welches Ausgabegerät bzw. welche Ausgabeform `apt-listchanges` benutzt. Möglich sind derzeit `pager`, `browser`, `xterm-pager`, `xterm-browser` (für die Darstellung in einem Textbrowser in ihrem Terminal), `text`, `mail` (Versand als Email), `gtk` (graphische Darstellung) und `none` (keine Ausgabe).

`--reverse`
Ausgabe der Änderungen in zeitlich umgekehrter Reihenfolge.

`--since=`*version*
Ausgabe aller Änderungen ab der angegebenen Version des Pakets.

`-v` (Langform `--verbose`)
Ausgabe in ausführlicher Form.

Im nachfolgenden Beispiel sehen Sie den Aufruf für das Paket *ruby-json*, wobei die Ausgabe als einfacher Text im Terminal erfolgt.

Aufrufbeispiel für apt-listchanges

```
# apt-listchanges -f text --which=both /var/cache/apt/archives/ruby-json_1.7.3-3_i386.deb
Lese Changelogs...
ruby-json (1.7.3-3) unstable; urgency=high

  * set urgency to high, as a security bug is fixed.
  * Add 10-fix-CVE-2013-0269.patch, adapted from upstream to fix denial of
    service and unsafe object creation vulnerability.
    [CVE-2013-0269] (Closes: #700436).

 -- Cédric Boutillier <cedric.boutillier@gmail.com>  Tue, 12 Feb 2013 23:14:48 +0100
...
#
```

32.3.4 Release-kritische Fehler mit `popbugs` finden

Mit dem Werkzeug `popbugs` aus dem Paket *debian-goodies* [Debian-Paket-debian-goodies] finden Sie release-kritische Fehler („release critical bugs“, abgekürzt mit *RC bugs*) in Paketen, die Sie hauptsächlich einsetzen. Dazu bezieht `popbugs` zunächst die Liste der *RC bugs* vom Debian BTS von der URL http://bugs.debian.org/release-critical/other/all.html und gleicht danach die gefundenen Fehler mit den Paketen ab, die auf ihrem System installiert sind. Beachten Sie daher, dass das Programm darauf aufbaut und für brauchbare Ergebnisse eine Internetverbindung benötigt. Die Grundlage des nachfolgenden, lokalen Vergleichs ist der allgemeine Installationsgrad der Debianpakete und die gesammelten Daten des *popularity contest*.

Abbildung 32.2 zeigt das Ergebnis der Recherche nach dem Aufruf von `popbugs` auf einem Debian 7 *Wheezy* in einem Browserfenster an (ohne Angabe von zusätzlichen Parametern etc.). Rufen Sie das Programm stattdessen über den Schalter `-o` gefolgt von einem Dateinamen zur Ausgabe auf, speichert `popbugs` die Auflistung als Datei unter dem angegebenen Pfad. Die Alternativschreibweise ist `--output=Ausgabedatei`.

Aufruf von popbugs und Ausgabe in die lokale Datei fehlerliste.html

```
$ popbugs -ofehlerliste.html
$
```

Sie erhalten jeweils eine direkte Verbindung zur Debian BTS mit einer paketweisen Auflistung. Weitere Informationen bekommen Sie, indem Sie in der Ausgabe auf einen mit einem Link hinterlegten Paketnamen klicken.

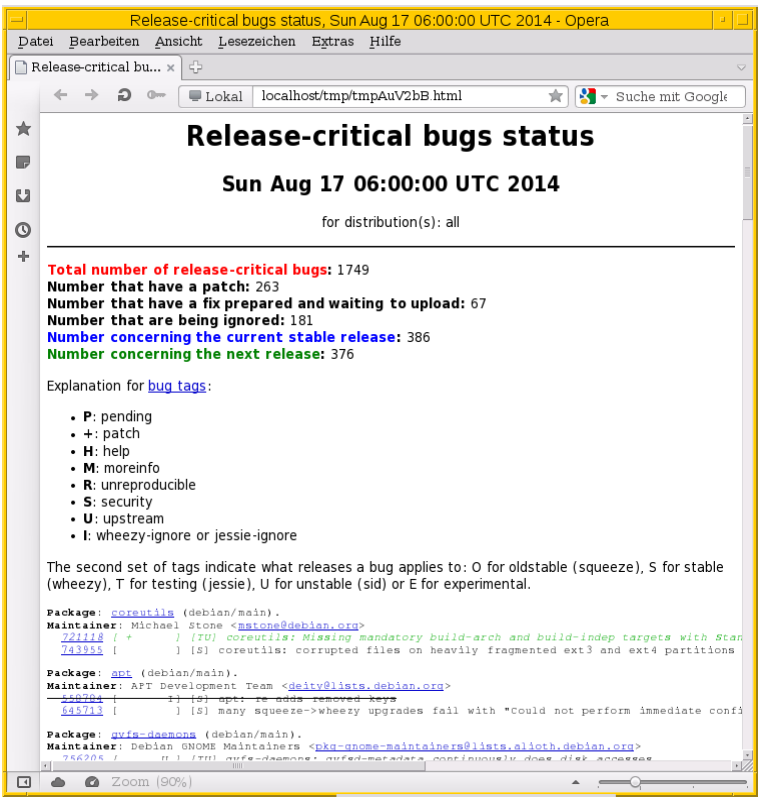

Abbildung 32.2: Auflistung der bekannten, kritischen Fehler nach der Analyse von `popbugs`

32.3.5 Release-kritische Fehler mit `rc-alert` finden

Das Werkzeug `rc-alert` aus dem Paket *devscripts* [Debian-Paket-devscripts] leitet seinen Namen aus den beiden Anfangsbuchstaben von *release critical* (dt. veröffentlichungskritisch) und dem Begriff *alert* (dt. Alarm, Warnung oder Meldung) ab. Inhaltlich folgt es dabei dem gleichen Ziel wie `popbugs` (siehe Abschnitt 32.3.4) und sucht dabei ebenfalls nach *RC bugs* vom Debian BTS über die URL http://bugs.debian.org/release-critical/other/all.html. Für brauchbare Ergebnisse benötigt es ebenso eine Internetverbindung.

`rc-alert` hebt sich von `popbugs` mit einigen Besonderheiten ab. Dazu gehört u.a. eine distributionsspezifische Suche und die Filterung der Paketliste anhand von Debtags (siehe Kapitel 13).

Nachfolgend sehen Sie die Arbeitsweise von `rc-alert` anhand des Pakets *apt*. Das Ergebnis umfasst neben der Nummer des Bugreports (*Bug*) dessen Titel (*Title*), die hinterlegten Flags (*Flags*) und die Veröffentlichung, für die der Fehler relevant ist (*Dists*). Der erste Bug gilt nur für Debian *stable*, während der zweite Bug für zukünftige Veröffentlichungen auf der Basis von Debian *testing* und *unstable* wichtig ist. Als Kennzeichnung für eine Veröffentlichung sind für Debian *oldstable* (O), *stable* (S), *testing* (T) und *unstable* (U) in Verwendung.

Einfache Suche nach RC bugs für das Paket apt mit Hilfe von rc-alert (Ausschnitt)

```
# rc-alert apt
Package: apt
Bug:     558784
Title:   apt: re-adds removed keys
Flags:   [        I] (lenny-ignore or squeeze-ignore)
Dists:   [S] (stable)

...

Package: apt
Bug:     776910
```

```
Title:   apt: upgrade from wheezy to jessie breaks in the middle
Flags:   [        ] (none)
Dists:   [TU] (testing, unstable)

#
```

Möchten Sie die **Recherche auf eine bestimmte Veröffentlichung einschränken**, helfen Ihnen dabei die beiden Schalter `-d` (Langform `--include-dists`) und `-o` (Langform `--include-dist-op`). Mit ersterem spezifizieren Sie über die o.g. Kennzeichnung die von Ihnen gewünschten Veröffentlichungen, so bspw. `--include-dists TUE` für Debian *testing unstable* und *experimental*. Damit erhalten Sie zunächst alle Fehler, die entweder in den Veröffentlchungen Debian *testing*, *unstable* oder *experimental* auftreten.

Suche über die Angabe der Veröffentlichung (Variante 1)

```
# rc-alert --include-dists TUE apt
Package: apt
Bug:     771428
Title:   apt tries to configure dbus before libdbus-1-3, fails to upgrade
Flags:   [        ] (none)
Dists:   [TUE] (testing, unstable, experimental)

Package: apt
Bug:     774924
Title:   apt: Jessie version cannot find upgrade path (but Wheezy version can)
Flags:   [ +      ] (patch)
Dists:   [E] (experimental)

Package: apt
Bug:     776910
Title:   apt: upgrade from wheezy to jessie breaks in the middle
Flags:   [        ] (none)
Dists:   [TU] (testing, unstable)
#
```

Mit dem zweiten Schalter legen Sie mittels `--include-dist-op and` fest, dass die Angabe der Veröffentlichung von `rc-bugs` als logisches *und* zu interpretieren ist. Damit begrenzen Sie die Ausgabe nur auf die angegebenen Veröffentlichungen.

Suche über die Angabe der Veröffentlichung (Variante 2)

```
# rc-alert --include-dists TUE --include-dist-op and apt
Package: apt
Bug:     771428
Title:   apt tries to configure dbus before libdbus-1-3, fails to upgrade
Flags:   [        ] (none)
Dists:   [TUE] (testing, unstable, experimental)
#
```

Wie eingangs angesprochen, gestattet Ihnen `rc-alert` auch eine **Suche anhand von Debtags**. Dazu bietet es den Schalter `--debtags` an. Für die Recherche nach allen Paketen, die Programmcode in der Programmiersprache Perl enthalten und als Plugin gekennzeichnet sind, gelingt Ihnen der folgende Aufruf:

Recherche zu Bugreports anhand der Debtags

```
# rc-alert --debtags implemented-in::perl,role::plugin
Package: dictionaries-common
Bug:     751367
Title:   unupgradeable: "shared/packages-wordlist doesn't exist"
Flags:   [        ] (none)
Dists:   [S] (stable)
Debtags: implemented-in::lisp, implemented-in::perl, role::plugin, role::program, scope:: ↩
    utility, works-with::dictionary
#
```

Für eine weitere Recherche mit ausführlicher Anleitung zu `rc-alert` empfehlen wir Ihnen den Beitrag von Gambaru [gambaru-rc-alert].

32.3.6 Welche der von mir genutzten Pakete benötigen Hilfe?

Das Programm `how-can-i-help` aus dem gleichnamigen Paket [Debian-Paket-how-can-i-help] geht genau dieser Frage nach und listet Ihnen auf, welche konkrete Art der Unterstützung ein Paket benötigt. Es ergänzt den Eintrag von `how-can-i-help` im Debian Wiki [Debian-Wiki-how-can-i-help] um die dazu passende Funktionalität auf der Kommandozeile. Obwohl es erst offiziell ab Debian 8 *Jessie* verfügbar ist, gibt es ebenso ein Paket aus *Debian Backports* für Debian 7 *Wheezy* (für *Debian Backports* siehe Kapitel 19).

Als Datenquelle für seine Aussagen nutzt es die Ultimate Debian Database (UDD) [Ultimate-Debian-Database]. Dabei handelt es sich um eine PostgreSQL-Datenbank, die `how-can-i-help` über eine webbasierte Schnittstelle abfragt. Beachten Sie daher, dass das Programm für brauchbare Ergebnisse eine Internetverbindung voraussetzt.

`how-can-i-help` integriert sich zudem über einen APT-Hook in APT und wird somit nach jeder Paketinstallation erneut ausgeführt. In Folge sehen Sie damit stets nur neu hinzugekommene Pakete, Fehler und den Bedarf an Hilfe. Ebenso können Sie das Programm über die Kommandozeile mit etlichen Schaltern aufrufen (Auswahl).

`-a` (Langform `--all`)
: Möglichkeiten für alle Pakete anzeigen, nicht nur für die installierten Pakete.

`-o` (Langform `--old`)
: Zeige nicht nur die neu hinzugekommenen Möglichkeiten oder die Möglichkeiten zu den zuletzt installierten Paketen, sondern auch die Möglichkeiten, welche bereits in der Vergangenheit aufgelistet wurden.

`-q` (Langform `--quiet`)
: Eine kompaktere Darstellung ohne Kopf- und Fußzeilen.

In Benutzung sind zudem eine Reihe von Abkürzungen für die konkrete Hilfe, die sich auf den Zustand eines Pakets beziehen. Diese Abkürzungen haben wir der Übersicht der *Work-Needing and Prospective Packages (WNPP)* entnommen [Debian-Wiki-WNPP]:

Orphaned (O)
: Kennzeichnung für ein verwaistes Paket, d.h. derzeit ohne Paketmaintainer.

Request For Adopter (RFA)
: Der derzeitige Paketmaintainer möchte die Verantwortung für das Paket abgeben und sucht einen Nachfolger.

Request For Help (RFH)
: Der derzeitige Paketmaintainer braucht generell Hilfe bei der Pflege des Pakets, z.B. in Form eines Co-Maintainers oder auch nur jemand, der Bugreports vorsortiert. Welche Hilfe sich der Paketbetreuer genau wünscht, finden Sie im angegebenen Bugreport.

Intent To Adopt (ITA)
: Jemand hat vor, das Paket zu übernehmen (Paketadoption). Diese Übernahme ist aber noch nicht geschehen.

Über die bereits bisher genannten Möglichkeiten zur Unterstützung haben sich die folgenden Wege bewährt, zur Weiterentwicklung und Verbesserung von Paketen beizutragen:

Request For Sponsorship (RFS)
: Jemand, der kein Debian-Entwickler ist, hat eine neue Version dieses Pakets vorbereitet und sucht einen Debian-Entwickler, der das Paket begutachtet und dann das Hochladen (den *Upload*) sponsort.

newcomer
: Pakete mit Bugreports, die mit der Markierung *newcomer* versehen sind. Eingeführt und früher bekannt unter der Markierung *gift* (engl. für *Geschenk*). Bugreports mit dieser Markierung gelten als „leichte Beute“ für Einsteiger in die Paketierung oder Entwicklung von Debian-eigenen Paketen. Diese Aufgaben sind i.d.R. ohne besondere Kenntnisse in der Paketierung lösbar, aber den Aufwand sollten Sie trotzdem nicht unterschätzen.

testing-autorm
: Kennzeichnung für Pakete, die in Kürze aus dem Zweig Debian *testing* entfernt werden. Hintergrund sind zumeist bisher nicht behobene, veröffentlichungskritische Fehler (*RC bugs*) des Pakets oder eines ihrer Abhängigkeiten.

no-testing
: Kennzeichnung für Pakete, die vor kurzem aus dem Zweig Debian *testing* entfernt wurden. Dies kann aufgrund bislang nicht behobener, veröffentlichungskritischer Fehler (*RC bugs*) passiert sein, aber auch weil der Paketbetreuer oder jemand anderes explizit die Entfernung des Pakets beantragt hat. Letzteres passiert häufiger, als Sie sich vorstellen können, z.B. wenn bei einer Bibliothek wegen einer `SONAME`-Änderung auch der Paketname korrigiert werden muss, bspw. von `libfoo7` zu `libfoo8`.

Über die letzten beiden Markierungen hat `how-can-i-help` eine Art Glaskugel-Funktion bzgl. zukünftiger Debian-Veröffentlichunge
Daraus ersehen Sie Tendenzen dahingehend, welche Pakete im Bestand bleiben. Das hat direkte Auswirkungen, welche Software zukünftig als Debian-Paket verfügbar bleibt und gepflegt wird. Haben Sie gerade ein Paket aus Debian *stable* installiert und `how-can-i-help` zeigt Ihnen im Anschluss zu ihrer Installation an, dass das Paket aus Debian *testing* herausgeflogen ist, klären Sie für sich, ob Sie längerfristig auf dieses Paket setzen wollen. Bei einer solchen Meldung ist die Chance groß, dass dieses Paket in der nächsten Veröffentlichung von Debian *stable* nicht mehr enthalten ist.

Was gibt es zu tun? (Ausschnitt)

```
$ how-can-i-help
New packages where help is needed, including orphaned ones (from WNPP):
 - apt-rdepends - https://bugs.debian.org/487125 - O (Orphaned)
 - ara - https://bugs.debian.org/450876 - O (Orphaned)
 - dctrl-tools - https://bugs.debian.org/768834 - O (Orphaned)
...

New packages removed from Debian 'testing' (the maintainer might need help):
 - apt-dpkg-ref - https://tracker.debian.org/pkg/apt-dpkg-ref
 - cpp-4.4 - https://tracker.debian.org/pkg/gcc-4.4
 - gcc-4.4 - https://tracker.debian.org/pkg/gcc-4.4
...
$
```

32.4 Auslaufende Sicherheitsaktualisierungen mit `check-support-status` anzeigen

Mit Hilfes des Werkzeugs `check-support-status` aus dem Paket *debian-security-support* [Debian-Paket-debian-security-support] lesen Sie die Informationen des Debian Security Teams [Debian-Security] im Terminal. Hilfreich ist es vor allem bei Installationen von Debian *oldstable*, da es Ihnen berichtet, für welche Pakete bereits jetzt schon bekannt ist, dass deren Security-Support in zukünftigen Veröffentlichungen nicht fortgesetzt wird.

Als Basis nutzt `check-support-status` zwei textbasierte Datenbanken, die sich unter `/usr/share/debian-security-support/security-support-ended` und `/usr/share/debian-security-support/security-support-limited` befinden. Erstere enthält die Pakete, deren Sicherheitsaktualisierungen enden, während die zweite Datei die Pakete auflistet, deren Sicherheitaktualisierungen lediglich eingeschränkt wird.

Abbildung 32.3: Auflistung der Paket ohne zukünftige Sicherheitsaktualisierungen nach der Analyse von `debian-security-support`

Aufruf über die Kommandozeile (Ausschnitt)

```
# check-support-status
Eingeschränkte Sicherheitsaktualisierungen für eines oder mehrere Pakete

Leider war es nötig, die Unterstützung von Sicherheitsaktualisierungen für
einige Pakete einzuschränken.

Davon sind die folgenden auf diesem System gefundenen Pakete betroffen:

* Quelle:webkit
  Einzelheiten: No security support upstream and backports not feasible, only for use on  ↩
      trusted content
  Betroffene Binärpakete:
  - libjavascriptcoregtk-1.0-0 (installierte Version: 1.8.1-3.4)
  - libjavascriptcoregtk-3.0-0 (installierte Version: 1.8.1-3.4)
  - libwebkitgtk-1.0-0 (installierte Version: 1.8.1-3.4)
  - libwebkitgtk-1.0-common (installierte Version: 1.8.1-3.4)
  - libwebkitgtk-3.0-0 (installierte Version: 1.8.1-3.4)
  - libwebkitgtk-3.0-common (installierte Version: 1.8.1-3.4)

...
#
```

`check-support-status` bietet die folgenden Schalter an (Auswahl):

`--list` *Dateiname*
: *Dateiname* bezeichnet eine Textdatei, in der die einzelnen Pakete aufgelistet sind. Bei Paketen, deren Sicherheitsaktualisierungen nicht fortgesetzt werden (`--type ended`), sind hier der Name des Quellpakets, die Versionsnummer der zuletzt unterstützten Variante, das Datum des Support-Endes sowie zusätzliche Informationen hinterlegt. Bei Paketen mit

eingeschränktem Support (`--type limited`) umfasst der Eintrag lediglich den Namen des Quellpakets und Zusatzinformationen. Ist keine Datei benannt, werden alle mitgelieferten Listen als Basis benutzt.

`--type` ***Variante***
bezeichnet die Art der Einschränkung der Sicherheitsaktualisierungen. Zur Auswahl stehen `ended` und `limited` für zukünftig endende bzw. eingeschränkte Sicherheitsaktualisierungen.

Kapitel 33

Versionierte Paketverwaltung

- Idee:
  - Verwaltung der Quellpakete als Versionskontrollsystem, bspw. Git

ToDo: entnommen/transferiert aus Paketformat im Detail: Aufbau und Format

Darüberhinaus bestehen experimentelle, (noch) nicht offiziell verwendete Varianten. Diese sollen es ermöglichen, als Basis für das Paket auch ein Versionskontrollsystem wie bspw. Git oder Bzr zu verwenden (siehe [Debian-Paket-dgit] und [Canonical-builder]).

Kapitel 34

Pakete und Patche datumsbezogen auswählen

Frage: Ist es möglich, die Patche bis zu einem ganz speziellen Datum einzuspielen?

Problem: Wir haben hier Entwicklungs-, Test- und Produktivumgebungen. Auf den Entwicklungsumgebungen werden immer die neuesten Patche eingespielt, das ist so gewünscht und wird gemacht. :) Das Problem ist auf den Test- und Produktivumgebungen. Es gibt bei uns die Anforderung, das wir Patche erst auf der Testumgebung installieren und testen und erst nach Freigabe der Patche auf der Produktivumgebung einspielen dürfen. Hier vergehen häufig 2-3 Wochen. Ich müsste also quasi heute auf der Testumgebung sagen, Patche einspielen und in 3 Wochen dann auf der Produktivumgebung, Patche bis zum 25.03.2013 15:36 einspielen. Alle neueren Patche müssten jetzt erst wieder auf die Testumgebung gespielt werden und neu freigegeben werden.

Gedanken zur Antwort:

- Problem tritt sehr häufig auf, bspw. in der Entwicklung. Ein Softwarestand wird zusammengestellt, ausführlich getestet und — falls alles gutgeht und freigegeben wurde — auf dem Produktivsystem ausgerollt.
- Variante 1 zur Lösung: Zustand des Testsystems mit allen Paketen und den Konfigurationsdateien wird gesichert/gemerkt, bspw. über eine Paketliste oder über Puppet, `rsync` o.ä.
- Variante 2: eine Art *Package Freeze*. Das Datum, bis zu dem noch Aktualisierungen von Paketen einfließen können, wird festgelegt. Das Zauberwort heißt http://snapshot.debian.org/ [Debian-Snapshots]. In der `/etc/apt/sources.list` stehen dann Einträge der Form:

```
deb http://snapshot.debian.org/archive/debian/20091004T111800Z/ lenny main
deb-src http://snapshot.debian.org/archive/debian/20091004T111800Z/ lenny main
deb http://snapshot.debian.org/archive/debian-security/20091004T121501Z/ lenny/updates main
deb-src http://snapshot.debian.org/archive/debian-security/20091004T121501Z/ lenny/updates ↩
    main
```

Der Zeitstring `20091004T121501Z` folgt der Form `yyyymmddThhmmssZ` oder vereinfacht `yyyymmdd`. Steht für ein angegebenes Datum kein Snapshot bereit, wird der zeitlich entsprechend vorherige ausgewählt.

Kapitel 35

Paketkonfiguration sichern

* wie sichere ich die bestehende Paketkonfiguration aus der Debconf-Datenbank * wie spiele ich das lokal wieder ein * Werkzeuge: **debconf-get-selections** debconf-set-selections

Kapitel 36

Paketverwaltung mit eingeschränkten Ressourcen für Embedded und Mobile Devices

Die Idee und Anregung für diese Thematik kommt dankenswerter Weise von Werner Heuser [Sentinel4Mobile], einem langjährigen Berliner Spezialisten für Linux auf mobilen Geräten. Den Hintergrund bildet die Frage, welche Programme und Methoden optimal für mobile Geräte sind, um einerseits eine möglichst lange Nutzungsdauer zu ermöglichen und andererseits dabei nur so viel Ressourcen zu verbrauchen, wie unbedingt erforderlich sind. Wir betrachten das daher nachfolgend anhand der Richtgrößen Batterielaufzeit, belegter Speicherplatz, Bildschirmgröße und die Erfordernisse an die CPU.

36.1 localepurge

Ist Speicherplatz auf dem Datenträger knapp, kann das Werkzeug `localepurge` [localepurge] weiterhelfen. Es steht Ihnen über die Paketverwaltung in dem gleichnamigen Debianpaket [Debian-Paket-localepurge] zur Verfügung.

`localepurge` entfernt alle Sprachpakete aus ihrer Installation, die nicht von Ihnen benötigt werden. Die Sprachpakete sind locales-Pakete und umfassen Übersetzungen der Sprachdateien und Handbücher (Manpages). Vor dem Aufräumen legen Sie fest, welche Sprachen Sie wirklich im Alltag benötigen — nicht immer gehören Japanisch oder Arabisch dazu. Die von Ihnen gewählte Einstellung speichert `localepurge` in der Datei `/etc/locale.nopurge`.

Nach der ersten Benutzung wird `localepurge` automatisch nach jedem Aufuf von APT ausgeführt, d.h. wenn Sie ein bereits genutztes Paket aktualisieren oder ein neues Paket hinzufügen. Bislang unklar ist, ob das auch für `aptitude` und die anderen Werkzeuge zur Paketverwaltung gilt. Im nachfolgenden Beispiel hat `localepurge` immerhin über 23 MB Platz geschaffen.

Entfernen nicht benötigter Sprachpakete mit localepurge

```
# localepurge
localepurge: processing locale files ...
   Purging /usr/share/locale/az
   Purging /usr/share/locale/bg
   Purging /usr/share/locale/ca
   Purging /usr/share/locale/cs
  ...
localepurge: Disk space freed in /usr/share/locale: 23528K
#
```

Nach unserer Beobachtung werkelt `localepurge` anstandsfrei. Wir betrachten es als hilfreich für Installationen, die selten modifiziert werden, auch wenn es etwas in Konflikt mit der Paketverwaltung steht:

- Bei der Aktualisierung und Deinstallation von Paketen verursacht die Benutzung von `localepurge` Fehlermeldungen, da in diesem Moment das Paket nicht mehr vollständig installiert ist. Die Überprüfung mittels `debsum` schlägt fehl (siehe dazu die Prüfung eines Pakets auf Veränderungen in Abschnitt 8.29).

- Das Programm bietet keine Möglichkeit an, die gelöschten Dateien wieder herzustellen. Daher bleibt Ihnen nur eine Neuinstallation aller bereinigten Pakete, für die Sie die entsprechenden Sprachdateien doch wieder benötigen (siehe dazu Pakete erneut installieren in Abschnitt 8.37).

Kapitel 37

Paketverwaltung ohne Internet

Bisher haben wir bei nahezu allen im Buch besprochenen Szenarien zumeist stillschweigend vorausgesetzt, dass die von Ihnen betreuten Rechnersysteme Zugriff auf das Internet haben. Die Informationen zu einem Debianpaket und auch das Paket selbst haben Sie entweder online recherchiert oder bereits von einem vertrauenswürdigen Spiegelserver bezogen.

Im administrativen Alltag treten jedoch auch Situationen auf, in denen Sie versuchen müssen, ohne einen Zugang zum Internet auszukommen. Diese Situationen sind vielleicht etwas beschwerlich, aber Klagen und Hilflosigkeit löst das Problem im Ernstfall meist nicht wirklich. Bei den nachfolgend beschriebenen Lösungswegen sind vorausschauendes Denken und Handeln von Vorteil.

37.1 Hintergrund und Einsatzfelder

Vollkommen berechtigt ist die Frage, ob eine Paketverwaltung heute überhaupt noch ohne Internet bzw. eine bestehende Netzwerkverbindung möglich ist. Wenn Sie etwas zurückdenken, erinnern Sie sich vielleicht daran, dass bis vor wenigen Jahren CDs und später DVDs und USB-Sticks als Installationsmedien üblich waren und ein Netzzugang eher die Ausnahme darstellte. Aufgrund der Allgegenwärtigkeit des Internets und der flächendeckenden Funkvernetzung ist die Situation heutzutage genau umgekehrt und eine netzbasierte Installation stellt den Standardfall dar. Nachfolgend sprechen wir die Fälle an, bei denen etwas Planung hilft, Momente ohne Netz zu überbrücken. Ob das ganze sinnvoll ist, ist projekt- und situationsabhängig. Wenn es sein muss, stellt sich diese Frage eigentlich nicht, sondern nur, ob und insbesondere mit welchen Mitteln Sie ein aufgetretenes Problem möglichst zeitnah lösen können.

Situationen, die hier zu berücksichtigen sind, umfassen bspw. Arbeiten von unterwegs aus (Zug oder Flugzeug), Reparaturen irgendwo in der unterversorgten Pampa, auf der Berghütte, auf einem Schiff ohne UMTS, in einem abgeschirmten Gebäude wie einem Bunker oder auch hinter einer recht restriktiven Firewall — bspw. in einem Hotel. Zu berücksichtigen sind außerdem teure Wählverbindungen via Satelit sowie autarke Geräte und Installationen, die möglichst verbrauchsarm zusammengestellt sind oder gar keinen permanenten Netzzugang haben *dürfen*. Dazu zählen bspw. Meßstationen von Sensornetzwerken in eher unzugänglichen Gebirgsregionen oder auf Gletschern.

Meist wird Ihnen die Situation erst dann bewusst, wenn der Bedarf entsteht. Es hilft daher, bereits im Vorfeld daran zu denken und vorauszuplanen, um zumindest darauf vorbereitet zu sein.

37.2 Strategien

Aus unserer Sicht bestehen mehrere, grundlegende Verfahren, die sich hier anbieten:

- die benötigten Pakete vorher explizit herunterladen
- die Einbindung fester Installationsmedien
- die Einbindung eines lokalen Paketmirrors

Diese drei Wege basieren auf Werkzeugen und Verfahren, die wir bislang im Buch bereits angesprochen haben. Sagen Ihnen diese nicht zu, werfen Sie einen genaueren Blick auf die Projekte `apt-offline`, `apt-zip`, `aptoncd` und Keryx, die wir im Anschluss unter Werkzeuge in Abschnitt 37.3 genauer vorstellen.

37.2.1 Benötigte Pakete vorher explizit herunterladen

Dieser Weg setzt voraus, dass Sie wissen, was Sie brauchen werden. Das gelingt nicht immer und lässt sich auch nicht in allen Fällen exakt vorhersagen. Bitte testen Sie das daher im Vorfeld aus.

Wenn jedoch feststeht, welche Pakete erforderlich sind, laden Sie diese zunächst explizit mittels APT oder `aptitude` herunter. Damit landen die neuen Pakete inklusive der zusätzlich benötigten Abhängigkeiten im lokalen Paketcache Ihres Systems. Zu einem späteren Zeitpunkt können Sie die dort hinterlegten Pakete hervorholen und auf Ihrem System installieren. Für das Paket *debtags* mittels APT sind bspw. diese beiden Aufrufe notwendig:

APT-Aufrufe zum Zwischenspeichern und späteren Installieren eines Pakets

```
# apt-get download debtags
# apt-get --no-download install debtags
#
```

Ausführlicher besprechen wir jeden der beiden Einzelschritte in Abschnitt 8.31 und Abschnitt 8.32. Das betrifft nicht nur APT, sondern auch `aptitude`.

37.2.2 Einbindung fester Installationsmedien

Vorbereitete Installationsmedien bekommen Sie von der Webseite des Debian-Projekts als ISO-Image und Live-CD bzw. -DVD [Debian-besorgen]. Alternativ können Sie auch selbst Diskimages erstellen und benutzen [Debian-Wiki-DiskImage]. Ein solches Installationsmedium binden Sie als lokales Paketarchiv wie folgt ein:

Einbindung eines lokalen CD/DVD-Images namens image.iso als loop device unter /mnt/iso

```
# mkdir /mnt/iso
# mount -o loop -t iso9660 image.iso /mnt/iso
#
```

Bitte beachten Sie dabei, dass der Paketbestand zwischen dem eingebundenen Diskimage und Ihrer lokalen Installation bezüglich Ihrer benötigten Architektur und der genutzten Veröffentlichung harmonieren muss. Passen beide nicht zusammen, provozieren Sie Versionskonflikte zwischen den beiden Paketbeständen und mit hoher Wahrscheinlichkeit werden Sie die benötigten Pakete von dem Diskimage nicht auf Ihr lokales System einspielen können.

Eine Alternative stellt das Werkzeug `apt-cdrom` dar. Dieses stellen wir Ihnen unter „Physische Installationsmedien mit `apt-cdrom` einbinden“ in Abschnitt 3.8 genauer vor.

37.2.3 Einbindung eines lokalen Paketmirrors

Variante Drei ist das Benutzen eines eigenen Paketmirrors. Dieser kann lokal vorliegen (siehe Kapitel 28), aber auch als mobile Kopie auf einer externen (USB-)Festplatte überall zum Einsatz kommen. Diesen Datenträger mounten Sie zunächst und tragen den Paketmirror danach als zusätzliche Paketquelle (*Repository*) in der Datei `/etc/apt/sources.list` auf Ihrem lokalen System ein. Ist der Datenträger bspw. als `/mnt/mirror` gemounted, sieht der Eintrag in der Liste der Paketquellen wie folgt aus:

Einbinden einer externen Festplatte als Paketmirror

```
deb file:/mnt/mirror/debian stable main contrib non-free
```

Fügen Sie nun Pakete zu Ihrem lokalen System hinzu, sucht die Paketverwaltung die Pakete auf der angegebenen Paketquelle und bezieht diese von dort.

Dieser Schritt setzt voraus, dass Sie über eine genügend großen, zusätzlichen, mobilen Datenträger verfügen, um einen solchen Paketmirror vorzubereiten (meist unproblematisch). Weiterhin probieren Sie zuvor aus, ob die Paketinstallation von diesem Paketmirror klappt, um im Ernstfall handlungsfähig zu sein (auch unproblematisch). Als drittes sind Sie angeraten, diesen Datenträger dann auch mitzunehmen und nicht im Büro vergessen (schon eher problematisch) — ansonsten ist alle Mühe umsonst.

37.3 Werkzeuge

37.3.1 Offline-Verwaltung mit *apt-get* und *wget*

In diesem Szenario kombinieren wir aus APT das Werkzeug `apt-get` mit `awk` und `wget`. Diese wunderbare Idee und Beschreibung ist entlehnt aus Frank Ronneburgs Debian-Anwenderhandbuch [Debian-Anwenderhandbuch-apt-offline] sowie dem Beitrag von Samuel Suter [Suter-apt-offline], hier jedoch nur für ausgewählte Pakete.

`awk` ist ein Analyse- und Filterprogramm für Textdaten [Debian-Paket-gawk]. `wget` ist hingegen ein Kommandozeilenprogramm, um Dateien über das Netzwerk zu beziehen [Debian-Paket-wget]. Während *gawk* zu den essentiellen Paketen zählt, ist *wget* optional und daher ihrerseits vor dessen Verwendung gegebenenfalls noch zu installieren.

Der Ablauf ist wie folgt:

1. Einhängen („mounten“) eines externen, mobilen Datenträgers im Verzeichnis `/medium`
2. Aktualisieren der lokalen Paketdatenbank mittels `apt-get update`
3. Erzeugen der URLs mittels `apt-get` für die Pakete, die zu aktualisieren sind, und Umleitung der Ausgabe in die lokale Datei `uris`
4. Generieren der passenden `wget`-Aufrufe aus den zuvor gespeicherten URLs mittels `awk`. Ausgabe auf `stdout` und Umleitung der Ausgabe in die lokale Skriptdatei `/medium/wget-script`.
5. Ausführung der der Skriptdatei zum Bezug der bezeichneten Pakete vom Spiegelserver und der Speicherung der bezogenen Pakete im lokalen Verzeichnis.

Aufrufreihenfolge mittels apt-get, awk und wget

```
# apt-get update
# apt-get -qq --print-uris dist-upgrade > uris
# awk '{print "wget -O " $2 " " $1}' < uris > /medium/wget-script
# cd /medium
# sh -x ./wget-script
```

Als Ergebnis dieser Aufrufe finden Sie auf dem mobilen Datenträger alle `deb`-Pakete, die zur Aktualisierung mittels `apt-get dist-upgrade` erforderlich sind. Unmounten Sie den mobilen Datenträger auf dem ersten System und mounten Sie diesen auf dem Zielsystem. Als finalen Schritt erfolgt auf dem Zielsystem ein Aufruf von `apt-get dist-upgrade` mit dem Paketcache auf dem mobilen Medium. Dazu verwenden Sie den Schalter `-o` mit der APT-Direktive `dir::cache::archives` und dem passenden Pfad zum Paketcache. Der Einfachheit halber heißt dieser hier ebenfalls `/medium`.

Eine Distributionsaktualisierung mit einem externen Paketcache

```
# apt-get -o dir::cache::archives="/medium/" dist-upgrade
```

37.3.2 Das Projekt *apt-offline*

- Projektseite [apt-offline-Projektseite]
- Debian-Paket *apt-offline* [Debian-Paket-apt-offline]
- GUI-Version zum Paket: *apt-offline-gui* [Debian-Paket-apt-offline-gui]
- setzt auf der Programmiersprache Python auf
- Beschreibungen: Offline Package Management for APT ([Ritesh-apt-offline], [xubuntu-apt-offline], [Damienoh-apt-offline])

- Ablauf:

  - Signatur für die Maschine erzeugen, die aktualisierte Paketinformationen und Pakete bekommen soll (auf der Maschine, die offline ist)

```
# apt-offline set /tmp/apt-offline.sig
```

- Option `--update`: nur Updates (Einspielen aktualisierter, fehlerbereinigter Pakete mit der gleichen Versionsnummer)
- Option `--upgrade`: nur Upgrades (Einspielen aktualisierter, fehlerbereinigter Pakete mit einer neueren Version)
- ohne Optionen: alles auf den allerneuesten Stand bringen (entspricht einem `dist-upgrade`)
  - Paketinfo für die offline-Maschine holen, inkl. Bug-Reports und Ausführung als parallele 5 Threads (auf einer Maschine, die über eine Netzanbindung verfügt)

```
# apt-offline get apt-offline.sig --bug-reports --threads 5
```

- bezogene Paketinformationen auf der offline-Maschine einspielen, hier beispielhaft als Datei von einem Speichermedium, welches über USB eingehängt ist (USB-Stick oder USB-Festplatte)

```
# apt-offline install /media/USB/apt-offline.zip
```

- aktualisiert den APT-Cache, sodaß alles daher kommt und nichts von extern bezogen werden muß

37.3.3 apt-zip

- *apt-zip* [Debian-Paket-apt-zip]
- baut ein fetch-Skript, welches die benötigten Pakete vom Mirror bezieht
  - die bezogenen Daten/Pakete können dann lokal eingespielt werden
- Beschreibung: http://svn.laiskiainen.org/apt-rpm/trunk/doc/offline.sgml
  - externer Datenträger als lokaler Paketcache für apt-get
  - Erzeugen eines wget-Skripts, welches die benötigten Pakete bezieht
- Hinweis:
  - apt-zip hat es nicht nach Jessie geschafft, sollten wir für die erste Ausgabe ignorieren.

37.3.4 Pakete mit `dpkg-split` aufteilen

- Werkzeug `dpkg-split` aus dem Paket *dpkg* [Debian-Paket-dpkg]
- zerlegt eine `deb`-Datei in kleinere Stücke und kann diese Stücke wieder zusammensetzen
- nützlich, um Binärpakete auf mehrere Medien mit geringer Kapazität zu verteilen (wenn man grade nichts anderes zur Hand hat . . .)
- Schalter (Auswahl):
  - `-s` (Langform: `--split`)
  - `-j` (Langform: `--join`)

37.3.5 aptoncd

- *aptoncd* [Debian-Paket-aptoncd]
- Erstellt Installations-CDs von Paketen
  - die vorher explizit mittels APT heruntergeladenen wurden
  - über den aktuellen Paketbestand (eine Art Schnappschuß) und baut daraus ein ISO-Image bzw. Repository mit allen aktuell genutzten Paketen
  - beschreibt sich selbst als Debian Packages Backup Tool
  - nutzbar als Installationsimage
  - setzt auf GNOME/GTK und Python 2.6 auf
    - seit längerem keine neuere Version verfügbar
    - letzte Release von 2007

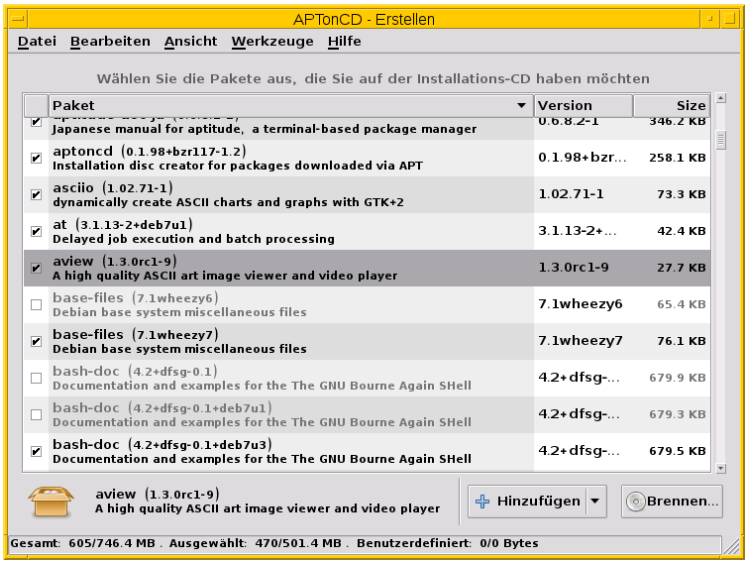

Abbildung 37.1: `aptoncd` im Einsatz

- Modi
  - Installations-CD erstellen
  - Dateien von der CD wieder herstellen
- Arbeitsweise
  - liest Paketdatenbank aus (Taste: F5)
  - erstellt Übersicht aus den verfügbaren und den installierten Paketen
    - Darstellung: 3 Spalten (Paketname+Version+Kurzbeschreibung, Version, Größe)
  - Darstellung der Gesamtmenge bzw. ausgewählt am unteren Fensterrand
  - erstellt Image daraus (Knopf: Brennen)
- Optionen und Parameter

`-c` (Langform `--packages-list=FILE`)
: `aptoncd` liest die Pakete aus der angegebenen Datei ein

`-l` (Langform `--cache-dir=PATH`)
Verwendet den angegebenen Pfad zum Paketcache anstatt dem Standardpfad `/var/cache/apt/archives`

`-r` (Langform `--temp-dir=PATH`)
verwendet den angegebenen Pfad als temporäres Verzeichnis, um Dateien zu kopieren und Images zu mounten

37.3.5.1 Keryx

- Projekt: [Keryx]
- „Keryx is a free, open source application for updating Linux. The Keryx Project started as a way for users with dialup, or low-bandwidth internet to be able to download and update packages on their debian based distribution of linux. Mainly built for Ubuntu, Keryx allows users to select packages to install, check for updates, and download these packages onto a USB portable storage device. The packages are saved onto the device and are then taken back to the Linux box that it originated from and are then installed.“
- „a portable, cross platform offline package manager for Debian based Linux distributions (Ubuntu and derivatives).“
- „Download software on computer connected to internet (it may be on Linux or Windows) and then install them to offline PC.“
- basiert auf dem Gimp Tool Kit
- letzte Veröffentlichung als Ubuntu PPA: 2011 (Version 1)
- (Stichwort Turnschuhnetzwerk, Sneakernet: https://de.wikipedia.org/wiki/Turnschuhnetzwerk)

Kapitel 38

Systeme mit schlechter Internet-Anbindung warten

Wir als Autoren und Systembetreuer haben uns mittlerweile an einen Internetzugang mit hoher Bandbreite gewöhnt. In der freien Wildbahn treffen wir aber durchaus auf Systeme, die mit weniger Bandbreite angebunden sind und auch gewartet werden möchten. Dazu gehört bspw. die Einwahl über ein Modem, via Integrated Services Digital Network (ISDN) oder eine volumenbeschränkte Internetanbindung. Auf mobilen Endgeräten erfolgt die Verbindung hingegen über GSM (2G, inkl. GPRS und EDGE), UMTS (3G), LTE (4G) oder auch Satellit und ist vom Standort und der Empfangsstärke abhängig.

Grob betrachtet, gibt es dabei drei Arten von Einschränkungen, mit denen Sie leben müssen:

- wackelige, unzuverlässige Verbindungen bei allen Funktechnologien,
- hohe Latenz bei Satelliten- und manchen Mobilfunkverbindungen und
- geringe Bandbreite bei einem Analog-Modem, bei ISDN und bei GSM-Verbindungen (2G)

Die Situation ist dabei durchaus ähnlich wie ohne Internet, nur mit wenigstens ein bisschen Internet.

Bei der Paketverwaltung stört am ehesten eine zu geringe Bandbreite. Latenz ist dabei nahezu irrelevant und bei instabilen Verbindungen stoßen Sie im Zweifelsfall den Download einfach nochmals an. Deswegen gehen wir im Folgenden vor allem auf Methoden zur Reduktion des Datenübertragungsvolumen ein.

Dabei spielen bei der Aktualisierung genau die Programme ihre Stärken aus, die nicht den gesamten, aktuellen Stand der Dinge übertragen, sondern nur die jeweiligen Änderungen. In das Rampenlicht rücken wir daher nachfolgend Konzepte, welche für die Paketlisten und die Paketinhalte nur die relevanten Differenzen herunterladen.

38.1 debdelta

Interessant für Sie ist das Werkzeug `debdelta-upgrade` aus dem Debdelta-Projekt ([Debdelta]). Es steht im Paket *debdelta* [Debian-Paket-debdelta] bereit, welches ebenfalls die drei weiteren Programme `debdelta`, `debdeltas` und `debpatch` enthält. Falls Sie das Werkzeug `cupt` installiert haben, ist `debdelta-upgrade` ebenfalls dort integriert (siehe Abschnitt 6.2.5).

`debdelta-upgrade` überträgt nicht die Pakete vollumfänglich, sondern nur die jeweiligen Unterschiede zwischen beiden Versionen — genannt *deltas*. Der Ideengeber für das in der Programmiersprache Python geschriebene Programm ist das Werkzeug `diff`, welches zeilenweise die Unterschiede zwischen zwei Dateien anzeigt.

In dem nachfolgenden Beispiel sehen wir einen `debdelta-upgrade`-Lauf mit einer virtuellen Übertragungsrate von 60kB/s über eine EDGE-Verbindung, deren Download-Raten zwischen 15kB/s und 25kB/s liegen. Das Ergebnis ist eine Verbesserung um den Faktor 2.5.

Optimiertes Herunterladen von Daten mit debdelta

```
# debdelta-upgrade
Created,    time  6.46sec, speed 31kB/sec, gir1.2-gtk-3.0_3.8.5-1_i386.deb
Created,    time  0.81sec, speed 75kB/sec, libgail-3-0_3.8.5-1_i386.deb
Created,    time  0.71sec, speed 82kB/sec, libgtk-3-bin_3.8.5-1_i386.deb
Created,    time  1.15sec, speed 72kB/sec, libio-socket-ssl-perl_1.955-1_all.deb
Created,    time  0.66sec, speed 15kB/sec, libmodule-build-tiny-perl_0.030-1_all.deb
Created,    time  3.36sec, speed 5672B/sec, libmodule-metadata-perl_1.000019-1_all.deb
Created,    time  0.72sec, speed 54kB/sec, libmodule-scandeps-perl_1.11-1_all.deb
Created,    time  0.51sec, speed 82kB/sec, libqt4-dbus_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time  3.82sec, speed 61kB/sec, libqt4-help_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time  8.99sec, speed 90kB/sec, libqt4-script_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time  3.09sec, speed 82kB/sec, libqt4-scripttools_4%3a4.8.5+git121-g2a9ea11+ ↩
    dfsg1-2_i386.deb
Created,    time  1.41sec, speed 94kB/sec, libqt4-sql_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time  0.58sec, speed 101kB/sec, libqt4-sql-sqlite_4%3a4.8.5+git121-g2a9ea11+ ↩
    dfsg1-2_i386.deb
Created,    time  1.52sec, speed 112kB/sec, libqt4-svg_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time  1.05sec, speed 90kB/sec, libqt4-test_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Delta is not present: libstring-flogger-perl_1.101242-1_1.101243-1_all.debdelta
Delta is not present: libsub-exporter-progressive-perl_0.001009-1_0.001010-1_all.debdelta
Created,    time 17.33sec, speed 59kB/sec, libqt4-xmlpatterns_4%3a4.8.5+git121-g2a9ea11+ ↩
    dfsg1-2_i386.deb
Created,    time  3.49sec, speed 61kB/sec, libqtdbus4_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time  1.38sec, speed 43kB/sec, librose-object-perl_0.860-1_all.deb
Created,    time  0.70sec, speed 17kB/sec, libstring-toidentifier-en-perl_0.11-1_all.deb
Created,    time  1.13sec, speed 43kB/sec, libsub-exporter-perl_0.986-1_all.deb
Created,    time  1.02sec, speed 14kB/sec, libsystem-command-perl_1.105-1_all.deb
Created,    time  0.62sec, speed 28kB/sec, libtest-file-perl_1.35-1_all.deb
Downloaded, time  0.42sec, speed 2580B/sec, libtext-hunspell-perl_2.05-1+b1_2.08-1_i386. ↩
    debdelta
Downloaded, time  0.49sec, speed 2764B/sec, libqt4-opengl_4%3a4.8.5+git121-g2a9ea11+dfsg-1 ↩
    _4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Created,    time  0.84sec, speed 45kB/sec, libtest-spec-perl_0.47-1_all.deb
Downloaded, time  0.38sec, speed 4031B/sec, librose-db-perl_0.769-1_0.771-1_all.debdelta
Created,    time  0.51sec, speed 41kB/sec, libxml-compile-cache-perl_0.995-1_all.deb
Downloaded, time  1.04sec, speed 1507B/sec, librt-client-rest-perl_1%3a0.43-2_1%3a0.44-1 ↩
    _all.debdelta
Created,    time  0.89sec, speed 72kB/sec, qdbus_4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386.deb
Downloaded, time  0.51sec, speed 3708B/sec, libavresample1_6%3a9.8-2+b2_6%3a9.10-1_i386. ↩
    debdelta
Created,    time  0.50sec, speed 85kB/sec, qt4-default_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Downloaded, time  0.46sec, speed 5070B/sec, libavdevice53_6%3a9.8-2+b2_6%3a9.10-1_i386. ↩
    debdelta
Downloaded, time  0.41sec, speed 5990B/sec, libavutil52_6%3a9.8-2+b2_6%3a9.10-1_i386. ↩
    debdelta
Downloaded, time  0.60sec, speed 7016B/sec, libqt4-network_4%3a4.8.5+git121-g2a9ea11+dfsg-1 ↩
    _4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Downloaded, time  0.62sec, speed  9kB/sec, libxml-compile-perl_1.35-1_1.39-1_all.debdelta
Downloaded, time  1.82sec, speed 7299B/sec, libswscale2_6%3a9.8-2+b2_6%3a9.10-1_i386. ↩
    debdelta
Downloaded, time  5.09sec, speed 2790B/sec, libyaml-libyaml-perl_0.38-3+b1_0.41-1_i386. ↩
    debdelta
Downloaded, time  2.18sec, speed  8kB/sec, python3-sip_4.14.7-4_4.15.2-2_i386.debdelta
```

```
Downloaded, time  1.90sec, speed 10kB/sec, libavfilter3_6%3a9.8-2+b2_6%3a9.10-1_i386. ↩
    debdelta
Downloaded, time  5.04sec, speed 3941B/sec, python-sip_4.14.7-4_4.15.2-2_i386.debdelta
Downloaded, time  1.82sec, speed 10kB/sec, libqt4-xml_4%3a4.8.5+git121-g2a9ea11+dfsg-1_4%3 ↩
    a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Downloaded, time  5.24sec, speed 3954B/sec, libqt4-declarative_4%3a4.8.5+git121-g2a9ea11+ ↩
    dfsg-1_4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Created,    time 31.47sec, speed 120kB/sec, qt4-dev-tools_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time 11.64sec, speed 72kB/sec, qt4-linguist-tools_4%3a4.8.5+git121-g2a9ea11+ ↩
    dfsg1-2_i386.deb
Created,    time  1.50sec, speed 86kB/sec, qt4-qtconfig_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Downloaded, time 21.87sec, speed 1090B/sec, libgtk-3-common_3.8.4-1_3.8.5-1_all.debdelta
Downloaded, time  6.28sec, speed 4569B/sec, libqtcore4_4%3a4.8.5+git121-g2a9ea11+dfsg-1_4%3 ↩
    a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Downloaded, time  4.94sec, speed 6927B/sec, libqt4-qt3support_4%3a4.8.5+git121-g2a9ea11+ ↩
    dfsg-1_4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Created,    time 15.00sec, speed 39kB/sec, qtcore4-l10n_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _all.deb
Created,    time  0.49sec, speed 38kB/sec, libtext-hunspell-perl_2.08-1_i386.deb
Downloaded, time  3.40sec, speed 11kB/sec, qt4-qmake_4%3a4.8.5+git121-g2a9ea11+dfsg-1_4%3a4 ↩
    .8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Downloaded, time  2.64sec, speed 16kB/sec, librose-db-object-perl_1%3a0.806-1_1%3a0.807-1 ↩
    _all.debdelta
Created,    time  4.11sec, speed 80kB/sec, libqt4-opengl_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time  1.84sec, speed 69kB/sec, librose-db-perl_0.771-1_all.deb
Downloaded, time  2.54sec, speed 18kB/sec, libqt4-dev-bin_4%3a4.8.5+git121-g2a9ea11+dfsg-1 ↩
    _4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Created,    time  1.74sec, speed 40kB/sec, librt-client-rest-perl_1%3a0.44-1_all.deb
Created,    time  1.20sec, speed 75kB/sec, libavresample1_6%3a9.10-1_i386.deb
Created,    time  0.79sec, speed 101kB/sec, libavdevice53_6%3a9.10-1_i386.deb
Created,    time  1.41sec, speed 89kB/sec, libavutil52_6%3a9.10-1_i386.deb
Downloaded, time  5.43sec, speed 14kB/sec, libqt4-dev_4%3a4.8.5+git121-g2a9ea11+dfsg-1_4%3 ↩
    a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Created,    time  8.74sec, speed 65kB/sec, libqt4-network_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Downloaded, time 13.47sec, speed 15kB/sec, gtk3-engines-oxygen_1.1.4-1_1.2.0-1_i386. ↩
    debdelta
Created,    time  9.57sec, speed 22kB/sec, libxml-compile-perl_1.39-1_all.deb
Created,    time  2.59sec, speed 56kB/sec, libswscale2_6%3a9.10-1_i386.deb
Created,    time  0.98sec, speed 64kB/sec, libyaml-libyaml-perl_0.41-1_i386.deb
Created,    time  1.08sec, speed 72kB/sec, python3-sip_4.15.2-2_i386.deb
Created,    time  2.00sec, speed 79kB/sec, libavfilter3_6%3a9.10-1_i386.deb
Created,    time  1.08sec, speed 73kB/sec, python-sip_4.15.2-2_i386.deb
Created,    time  1.10sec, speed 116kB/sec, libqt4-xml_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Downloaded, time 14.62sec, speed 15kB/sec, libqt4-designer_4%3a4.8.5+git121-g2a9ea11+dfsg-1 ↩
    _4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Created,    time 20.09sec, speed 54kB/sec, libqt4-declarative_4%3a4.8.5+git121-g2a9ea11+ ↩
    dfsg1-2_i386.deb
 Error: applying of delta for libgtk-3-common failed:  : Error, 220 locale files are absent ↩
     . (non retriable)
Downloaded, time 25.98sec, speed 11kB/sec, libavformat54_6%3a9.8-2+b2_6%3a9.10-1_i386. ↩
    debdelta
Created,    time 18.78sec, speed 82kB/sec, libqtcore4_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time 14.29sec, speed 73kB/sec, libqt4-qt3support_4%3a4.8.5+git121-g2a9ea11+ ↩
    dfsg1-2_i386.deb
Downloaded, time 36.54sec, speed 15kB/sec, libav-tools_6%3a9.8-2+b2_6%3a9.10-1_i386. ↩
    debdelta
```

```
Created,    time 26.96sec, speed 45kB/sec, qt4-qmake_4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386 ↩
    .deb
Created,    time 23.77sec, speed 22kB/sec, librose-db-object-perl_1%3a0.807-1_all.deb
Downloaded, time 54.52sec, speed 12kB/sec, libgtk-3-0_3.8.4-1_3.8.5-1_i386.debdelta
Created,    time 36.84sec, speed 43kB/sec, libqt4-dev-bin_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Downloaded, time 54.63sec, speed 14kB/sec, libavcodec54_6%3a9.8-2+b2_6%3a9.10-1_i386. ↩
    debdelta
Created,    time 58.14sec, speed 14kB/sec, libqt4-dev_4%3a4.8.5+git121-g2a9ea11+dfsg1-2 ↩
    _i386.deb
Created,    time  4.72sec, speed 70kB/sec, gtk3-engines-oxygen_1.2.0-1_i386.deb
Downloaded, time 48.70sec, speed 16kB/sec, libqtgui4_4%3a4.8.5+git121-g2a9ea11+dfsg-1_4%3a4 ↩
    .8.5+git121-g2a9ea11+dfsg1-2_i386.debdelta
Created,    time 25.46sec, speed 140kB/sec, libqt4-designer_4%3a4.8.5+git121-g2a9ea11+dfsg1 ↩
    -2_i386.deb
Created,    time 10.35sec, speed 63kB/sec, libavformat54_6%3a9.10-1_i386.deb
Created,    time 40.21sec, speed 80kB/sec, libav-tools_6%3a9.10-1_i386.deb
Created,    time 19.10sec, speed 93kB/sec, libgtk-3-0_3.8.5-1_i386.deb
Created,    time 59.42sec, speed 48kB/sec, libavcodec54_6%3a9.10-1_i386.deb
Created,    time 58.16sec, speed 69kB/sec, libqtgui4_4%3a4.8.5+git121-g2a9ea11+dfsg1-2_i386 ↩
    .deb
Downloaded, time 228.14sec, speed 12kB/sec, libgtk-3-common_3.8.5-1_all.deb
Delta-upgrade statistics:
 total resulting debs, size 37MB time 637sec virtual speed 60kB/sec
Sorry, no forensic logs were generated
debdelta-upgrade  331.64s user 50.87s system 56% cpu 11:15.57 total
#
```

38.2 PDiffs

PDiffs ist die Abkürzung für *Package list diff* und eine sehr nützliche Option von APT. Diese Option bewirkt, dass nicht mehr die gesamte aktuelle Paketliste zu ihrem System übertragen wird, sondern nur noch die Änderungen zum aktuellen Stand. Da diese Änderungen überschaubar sind, spart das a) Zeit, b) genutzte Bandbreite und c) schont ihren Geldbeutel.

Vor deren Benutzung schalten Sie diese Option explizit in der Konfiguration von APT ein, bspw. über einen Eintrag in der Datei `/etc/apt/apt.conf`:

Konfigurationseintrag für PDiffs

```
Acquire::PDiffs "true";
```

Einsatzfeld der Option
Bitte beachten Sie, dass diese Option nur für die Debian-Veröffentlichungen *testing*, *experimental* und *Sid* besteht. Für die stabile Veröffentlichung hat die Option keinen Effekt.

Kapitel 39

Der APT- und `aptitude`-Wunschzettel

Es existieren eine Reihe von Funktionen, die wir in APT und `aptitude` vermissen, aber die es schon woanders gibt:

- `apt-get`: ausgewählte Paketquellen aktualisieren — geht bereits mit SmartPM (Abschnitt 6.4.3)

Teil IV

Ausblick

Kapitel 40

Notizen

TODO: Wollen wir hier ein wenig über die Zukunft von Paketmanagement unter Debian orakeln? CUPT könnte man hier nochmals erwähnen. Das web-basierte Zeugs vielleicht auch (Stichworte Tablets und Smartphones).

- welche Fragen blieben bislang offen und unbeantwortet
- hat Paketmanagement — so wie es jetzt vorliegt und verwendet wird — überhaupt eine Zukunft
  - Modell und Struktur wird langsam volljährig
  - falls ja, was spricht denn dafür
  - falls nein, was wäre denn eine geeignete Alternative dazu
    - gibt es vllt. schon kreative Ansätze, die langsam vor sich hin köcheln
    - was ist mit den Containerformaten wie Flatpak, Open Container Format (OCF), App Container Image (ACI) und Docker?
- welche Punkte sind zu verbessern bzw. zu ergänzen
  - an welchen Punkten hakelt es immer wieder
  - hat das Modell Risiken und Schwachstellen
  - was wird derzeit nicht abgedeckt
  - Anwendungsfälle
- was sind die häufigsten Vorkommnisse, an denen etwas schiefgeht
- welche der im Buch vorgestellten Ansätze und Lösungen haben Potential für die Zukunft

Kapitel 41

Pakete selber bauen

- TODO: Evtl. woanders hin?
- Beispiel: anhand des deb-Pakets für das vorliegende Buch
  - weil (1): ist komplett
  - weil (2): ist in den Quellen für das Buch gleich dabei
  - weil (3): hängt von keinen weiteren Paketen ab
  - weil (4): ist überschaubar
  - weil (5): damit können wir testen, ob wir unser Paket für das Buch richtig bauen bzw. ob noch etwas fehlt

Kapitel 42

Ein eigenes Debian-Repository aufbauen

- siehe dazu: Michael Stapelberg: Kurz-Howto: Eigenes Debian-Repository aufbauen [Stapelberg-Debian-Repo]

Kapitel 43

Zukunft von APT, dpkg und Freunden

TODO: Drei-Weg-Konfigdatei-Mergen

- Libelektra [libelektra] ist IIRC eine generische Konfigdatei-Parser-Bibliothek und
- Cleverer Config merge [Bean-clever-merge-config]

Kapitel 44

Fazit / Zusammenfassung

TODO: Mehr und Aufsplitten

44.1 Empfehlungen für Einsteiger

44.1.1 Mit welchem Programm zur Paketverwaltung soll ich anfangen?

Das hängt sehr davon ab, welche Art von Benutzerschnittstelle Sie bevorzugen — oder auch, in welcher Situation (Desktop, Server via ssh) Sie gerade sind:

Kommandozeile
: `apt`. Es ist einfach und intuitiv, schnell getippt, hat angenehme farbliche Hervorhebungen in der Ausgabe und kann praktisch alles, womit Sie im Alltag zu tun haben. Bei älteren Veröffentlichungen ohne `apt` als Befehl weichen Sie auf die Befehle `apt-get` und `apt-cache` aus, aber die Nachteile sind nur geringfügig. Im wesentlichen umfaßt das mehr Tippen, je nach Aufgabe das passende Programm auswählen und keine Farbe in der Standard-Einstellung.

Interaktiver Textmodus
: `aptitude`. Nicht nur, weil es auch das einzige Programm mit dieser Benutzerschnittstelle ist, sondern auch, weil es einem sehr viele Informationen und Möglichkeiten bietet. Diese Darstellung in der Gesamtheit auf der Kommandozeile zu erreichen, gelingt nur mit einem Dutzend parallel geöffneter Terminalfenster.

Graphische Benutzerschnittstelle
: Synaptic (siehe Abschnitt 6.4.1). Es bildet unserer Meinung nach das Debian-Paketsystem am besten ab und reduziert es nicht auf einen „App-Store“.

Anhang A

Bibliography

- [alien] Joey Hess: Projektseite zu *alien*, http://joeyh.name/code/alien/
- [Allbery-Debian-Popularity] Debian and popularity, Blog-Posting von Russ Allbery, http://www.eyrie.org/~eagle/journal/-2012-01/004.html
- [Amberg-Linux-Server-Praxishandbuch] Eric Amberg: Linux-Server mit Debian 6 GNU/Linux: Das umfassende Praxishandbuch — Aktuell für die Versionen Debian 6.0 (Squeeze) und Debian 5.0 (Lenny), mitp; Auflage: 2012 (24. April 2012), Sprache: Deutsch, ISBN-10: 3826655346, ISBN-13: 978-3826655340
- [Aoki-Debian-Referenz] Osamu Aoki: Debian Referenz, deutsche Übersetzung, http://www.debian.org/doc/manuals/debian-reference/index.de.html
- [Aoki-Debian-Referenz-Mirror] Osamu Aoki: Debian Referenz, deutsche Übersetzung, http://qref.sourceforge.net/quick/index.de.htm
- [apper] Apper, http://kde-apps.org/content/show.php/Apper?content=84745
- [appnr] Appnr, http://appnr.com/
- [apt2] APT 2.0, http://wiki.debian.org/Apt2
- [apt4rpm] `apt4rpm`: http://apt4rpm.sourceforge.net/
- [aptitude-categorical-browser-to-be-removed] Kategorie-Browser in Aptitude wird voraussichtlich entfernt, https://bugs.debian.org/-cgi-bin/bugreport.cgi?bug=686124#44
- [aptitude-dokumentation] Dokumentation zu `aptitude` in allen bisher verfügbaren Sprachen, http://aptitude.alioth.debian.org/-doc/
- [aptitude-dokumentation-package-list] Dokumentation zu `aptitude`: Darstellung der Paketliste, http://aptitude.alioth.debian.org/-doc/en/ch02s05s01.html
- [aptitude-dokumentation-paketdarstellung] Dokumentation zu `aptitude`: Paketdarstellung, http://aptitude.alioth.debian.org/-doc/en/ch02s05s01.html#secDisplayFormat
- [aptitude-dokumentation-text-colors-and-styles] Dokumentation zu `aptitude`: Text colors and Styles, http://aptitude.alioth.debian.or doc/en/ch02s05s03.html
- [aptitude-dokumentation-themes] Dokumentation zu `aptitude`: Themes, http://aptitude.alioth.debian.org/doc/en/ch02s05s06.html
- [aptitude-dokumentation-veraltet] veraltete Dokumentation zu `aptitude`, http://www.algebraicthunk.net/~dburrows/projects/-aptitude/doc/
- [apt-Salve] Ravi Salve: 25 Useful Basic Commands of APT-GET and APT-CACHE for Package Management, http://www.tecmint.com useful-basic-commands-of-apt-get-and-apt-cache-for-package-management/
- [Aptosid] Aptosid, http://aptosidusers.de/

- [aptsh-Projekt] `aptsh`-Projektseite bei BerliOS, http://developer.berlios.de/projects/aptsh/
- [apt-cacher-ng-Projektseite] Projektseite zu *apt-cacher-ng*, http://www.unix-ag.uni-kl.de/~bloch/acng/
- [apt-dater-Projektseite] Projektseite zu `apt-dater`, http://www.ibh.de/apt-dater/
- [apt-fast] Projektseite zu `apt-fast` auf Github, https://github.com/ilikenwf/apt-fast
- [apt-get.org] Suche nach inoffiziellen Debian-Paketen, http://www.apt-get.org/
- [apt-get-update-bug-764503] Debian Bug Report 764503, https://bugs.debian.org/764503
- [apt-mirror-Projektseite] Projektseite zu *apt-mirror*, http://apt-mirror.github.io/
- [apt-mirror-ubuntu] Felipe Alfaro Solana: Create a local mirror of Ubuntu packages using `apt-mirror`, http://blog.felipe-alfaro.com/2014/05/30/create-a-local-mirror-of-ubuntu-packages-using-apt-mirror/
- [apt-mirror-ubuntu2] Create a Local Ubuntu Repository using Apt-Mirror and Apt-Cacher, Packt Publishing, https://www.packtpub.com/books/content/create-local-ubuntu-repository-using-apt-mirror-and-apt-cacher
- [apt-offline-Projektseite] Projektseite zu *apt-offline*, http://apt-offline.alioth.debian.org/
- [APT-Repo-PostgreSQL] APT-Repositories von PostgreSQL, https://wiki.postgresql.org/wiki/Apt
- [APT-Repo-X2Go] APT-Repositories von X2Go, http://wiki.x2go.org/doku.php/wiki:repositories:debian
- [apt-rpm] `apt-rpm`, http://apt-rpm.org/
- [Arch-Linux-pacapt] `pacapt` auf GitHub: https://github.com/icy/pacapt
- [AsciiDoc] AsciiDoc, Text-basierte Dokumenten-Generierung, http://www.methods.co.nz/asciidoc/
- [awk] Al Aho, Brian Kernighan, und Peter Weinberger: The AWK Programming Language, Addison-Wesley, 1988, ISBN 0-201-07981-X
- [backports.org-moved-to-backports.debian.org] backports.org moved to backports.debian.org, http://backports.debian.org/news/-backports.org_moved_to_backports.debian.org/
- [Backports-Mailingliste] Backports-Mailingliste, https://lists.debian.org/debian-backports/
- [BackTrack] BackTrack, http://www.backtrack-linux.org/
- [Bailey-Maximum-RPM] Edward C. Bailey: Maximum RPM. Taking the Red Hat Package Manager to the Limit, Red Hat Inc., 2000, ISBN 1-888172-78-9, http://www.rpm.org/max-rpm/
- [Bailey-Maximum-RPM-verify] When Verification Fails — rpm -V Output in Edward C. Bailey: Maximum RPM. Taking the Red Hat Package Manager to the Limit, Red Hat Inc., 2000, ISBN 1-888172-78-9, http://www.rpm.org/max-rpm/
- [Bean-clever-merge-config] Jules Bean: Cleverer conffile merge, https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=32877
- [Beckert-Aptitude-Textmodus] Axel Beckert: Aptitude. Pakete komfortabel im Text-Modus verwalten, Vortrag im Rahmen des Linuxday Dornbirn, November 2013, http://fileshare.lugv.at/public/vortraege2013/linuxday/aptitude-linuxday.html
- [Beckert-Blog] Axel Beckerts Blog, http://noone.org/blog
- [Beckert-Blog-Aptitude-Gtk-Will-Vanish] Axel Beckert: `aptitude-gtk` will likely vanish, http://noone.org/blog/English/-Computer/Debian/aptitude-gtk%20will%20likely%20vanish.futile
- [Beckert-Blog-Finding-Libraries] Axel Beckert: Finding libraries not marked as automatically installed with `aptitude`, http://noone.org/blog/English/Computer/Debian/CoolTools/Finding%20libraries%20not%20marked%20as%20automatically%20inst
- [Beckert-Blog-Hardlinking-Duplicate-Files] Axel Beckert: Automatically hardlinking duplicate files under `/usr/share/doc` with APT, http://noone.org/blog/English/Computer/Debian/Automatically%20hardlinking%20duplicate%20documentation%20f
- [Beckert-Hofmann-Aptitude-1-LinuxUser] Axel Beckert, Frank Hofmann: Dynamisches Duo. Apt-get und Aptitude (Teil 1), LinuxUser 04/2013, S. 90ff., http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2013/04/Apt-get-und-Aptitude-Teil-1

- [Beckert-Hofmann-Aptitude-2-LinuxUser] Axel Beckert, Frank Hofmann: Paarweise. Apt-get und Aptitude (Teil 2), LinuxUser 06/2013, S. 90ff., http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2013/06/Apt-Get-und-Aptitude-im-Einsatz
- [Beckert-Webseite] Axel Beckerts Webseite, http://axel.beckert.ch/
- [biarch] Fedora Project: Biarch Spin, http://fedoraproject.org/wiki/Biarch_Spin
- [Buero2.0] Büro 2.0, Berlin, http://www.buero20.org/
- [bugs-found-during-book-writing] Bugs, die während dem bzw. durch das Schreiben dieses Buches gefunden wurden, https://bugs.debi cgi-bin/pkgreport.cgi?tag=found-during-book-writing;users=abe%40debian.org
- [Canonical-builder] Canonical: builder - Construct a branch from a recipe, http://doc.bazaar.canonical.com/plugins/en/builder-plugin.html
- [checkinstall] Projektseite zu *checkinstall*, http://asic-linux.com.mx/~izto/checkinstall/
- [Click-Paket-Format] Canonicals *Click* Paketformat, https://click.readthedocs.org/en/latest/
- [Click-Paket-Format-Diskussionen] Can Ubuntu Click Address Linus Torvalds' Binary Problems?, https://www.linux.com/-news/software/applications/799449-can-ubuntu-click-address-linus-torvalds-binary-problems/
- [CLT] Chemnitzer Linux-Tage, http://chemnitzer.linux-tage.de/
- [Communtu] Webseite des Communtu-Projekts, http://de.communtu.org/
- [CreativeCommons] Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 4.0 International Lizenz, http://creativecommons.org/licenses/by-sa/4.0/
- [Cupt-Tutorial] Cupt Tutorial, http://people.debian.org/~jackyf/cupt2/tutorial.html
- [curses-apt-key] curses-apt-key, https://github.com/xtaran/curses-apt-key
- [curses-apt-key-braucht-gui-apt-key-aufsplittung] Aufsplittung von gui-apt-key in Bibliothek und Frontend gewünscht, https://bugs.del 675199
- [curses-apt-key-itp] Intent to package curses-apt-key, https://bugs.debian.org/675187
- [Damienoh-apt-offline] Damien Oh: How to Update/Upgrade Your Ubuntu Without Internet Connection, http://www.maketecheasier.c update-upgrade-ubuntu-without-internet-connection/
- [DamnSmallLinux] Damn Small Linux, http://www.damnsmalllinux.org/
- [DebConf] Debian Entwicklerkonferenz (DebConf), http://www.debconf.org/
- [DebConf5] Debian Entwicklerkonferenz (DebConf) in Helsinki, http://debconf5.debconf.org/
- [Debdelta] Debdelta, Pakete als Differenzen zur vorherigen Paket-Version, http://debdelta.debian.net/
- [DebianDerivativeCensus] Debian-Derivate-Zensus, http://wiki.debian.org/Derivatives/Census
- [Debianforum-Wiki-Backports] Debian Backports im Debianforum Wiki: http://wiki.debianforum.de/Backports
- [DebianLiveSystem] The Debian Live Systems project, http://live.debian.net/
- [Debian-Anwenderhandbuch] Frank Ronneburg: Das Debiananwenderhandbuch, http://debiananwenderhandbuch.de/
- [Debian-Anwenderhandbuch-apt-offline] Frank Ronneburg: Das Debiananwenderhandbuch, APT offline benutzen, http://debiananwen apt-offline.html
- [Debian-Anwenderhandbuch-apt-optionen] Frank Ronneburg: Das Debiananwenderhandbuch, Die Optionen von APT, http://debianan apt-get.html
- [Debian-Architekturen] Liste der von Debian unterstützten Architekturen, http://www.debian.org/ports/
- [Debian-Archive] Archiv der von Debian nicht mehr unterstützten Veröffentlichungen, http://archive.debian.org/

- [Debian-Backports] Debian Backports: http://backports-master.debian.org/
- [Debian-besorgen] Debian besorgen. Installationsmedien und ISO-Images auf der Debian-Webseite, http://www.debian.org/-distrib/
- [Debian-Bug-Tracking-System] Debian Bug Tracking System (Debian BTS), https://www.debian.org/Bugs/
- [Debian-DebSrc3.0] Projects DebSrc3.0, http://wiki.debian.org/Projects/DebSrc3.0
- [Debian-Debtags] Debtags Projekt, http://debtags.debian.org/
- [Debian-Debtags-Old] Debtags Projekt, http://debtags.debian.net/
- [Debian-Debtags-Editor] Debtags Editor, http://debtags.debian.net/edit/
- [Debian-Debtags-Search] Debtags Projekt, Suche, http://debtags.debian.org/search
- [Debian-Debtags-Search-By-Tags] Debtags Projekt, Suche anhand der Schlagworte, http://debtags.debian.org/search/bytag
- [Debian-Debtags-Statistics] Debtags Projekt, Statistische Daten, http://debtags.debian.org/reports/stats/
- [Debian-Developers-Reference] Developer's Reference Team: Debian Developer's Reference, deutsche Übersetzung, http://www.debia doc/manuals/developers-reference/index.html
- [Debian-Donations] Spenden an Debian, http://www.debian.org/donations
- [Debian-History] Debian Documentation Team: A Brief History of Debian, Chapter 3, Debian Releases, http://www.debian.org/-doc/manuals/project-history/ch-releases.de.html
- [Debian-Manpages] Debian Man Page Lookup, http://http://manpages.debian.org/
- [Debian-Mirror-Checker] Debian Mirror Checker, http://mirror.debian.org/status.html
- [Debian-Mirror-Doku] Dokumentation zur Auswahl eines Netzwerk-Spiegel-Servers, http://www.debian.org/releases/stable/-i386/ch06s03.html#apt-setup-mirror-selection
- [Debian-Package-Basics] What is a Debian package? http://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.en.html
- [Debian-Paketliste] Debian-Webseite, Paketliste, https://packages.debian.org/de/jessie/
- [Debian-Paketsuche] Debian-Webseite, Paketsuche, https://www.debian.org/distrib/packages#search_contents
- [Debian-Paket-adept] Debian-Paket *adept*, http://packages.debian.org/adept
- [Debian-Paket-adequate] Debian-Paket *adequate*, http://packages.debian.org/de/jessie/adequate
- [Debian-Paket-alien] Debian-Paket *alien*, http://packages.debian.org/de/stable/alien
- [Debian-Paket-apper] Debian-Paket *apper*, https://packages.debian.org/jessie/apper
- [Debian-Paket-approx] Debian-Paket *approx*, http://packages.debian.org/de/stable/approx
- [Debian-Paket-apt] Debian-Paket *apt*, https://packages.debian.org/de/stable/apt
- [Debian-Paket-aptoncd] Debian-Paket *aptoncd*, http://packages.debian.org/de/stable/aptoncd
- [Debian-Paket-apt-cacher] Debian-Paket *apt-cacher*, http://packages.debian.org/de/stable/apt-cacher
- [Debian-Paket-apt-cacher-ng] Debian-Paket *apt-cacher-ng*, http://packages.debian.org/de/stable/apt-cacher-ng
- [Debian-Paket-apt-cdrom-setup] Debian-Paket *apt-cdrom-setup*, http://packages.debian.org/de/stable/apt-cdrom-setup
- [Debian-Paket-apt-dater] Debian-Paket *apt-dater*, https://packages.debian.org/de/stable/apt-dater
- [Debian-Paket-apt-dpkg-ref] Debian-Paket *apt-dpkg-ref*, http://packages.debian.org/de/stable/apt-dpkg-ref
- [Debian-Paket-apt-doc] Debian-Paket *apt-doc*, http://packages.debian.org/de/stable/apt-doc

- [Debian-Paket-apt-fast] Debian-Paket *apt-fast*, http://packages.debian.org/de/stable/apt-fast
- [Debian-Paket-apt-listbugs] Debian-Paket *apt-listbugs*, https://packages.debian.org/de/stable/apt-listbugs
- [Debian-Paket-apt-listchanges] Debian-Paket *apt-listchanges*, https://packages.debian.org/de/stable/apt-listchanges
- [Debian-Paket-apt-mirror] Debian-Paket *apt-mirror*, https://packages.debian.org/de/stable/apt-mirror
- [Debian-Paket-apt-move] Debian-Paket *apt-move*, https://packages.debian.org/de/stable/apt-move
- [Debian-Paket-apt-offline] Debian-Paket *apt-offline*, http://packages.debian.org/de/stable/apt-offline
- [Debian-Paket-apt-offline-gui] Debian-Paket *apt-offline-gui*, http://packages.debian.org/de/stable/apt-offline-gui
- [Debian-Paket-apt-rdepends] Debian-Paket *apt-rdepends*, http://packages.debian.org/de/stable/apt-rdepends
- [Debian-Paket-apt-setup] *apt-setup*, http://packages.debian.org/de/stable/apt-setup-udeb
- [Debian-Paket-apt-show-versions] Debian-Paket *apt-show-versions*, http://packages.debian.org/de/stable/apt-show-versions
- [Debian-Paket-apt-spy] Debian-Paket *apt-spy*, http://packages.debian.org/de/stable/apt-spy
- [Debian-Paket-apt-transport-debtorrent] Debian-Paket *apt-transport-debtorrent*, http://packages.debian.org/de/stable/apt-transport-debtorrent
- [Debian-Paket-apt-zip] Debian-Paket *apt-zip*, http://packages.debian.org/de/stable/apt-zip
- [Debian-Paket-ara] Debian-Paket *ara*, http://packages.debian.org/de/stable/ara
- [Debian-Paket-aria2] Debian-Paket *aria2*, http://packages.debian.org/de/stable/aria2
- [Debian-Paket-autopkgtest] Debian-Paket *autopkgtest*, https://packages.debian.org/de/stable/autopkgtest
- [Debian-Paket-auto-apt] Debian-Paket *auto-apt*, https://packages.debian.org/de/stable/auto-apt
- [Debian-Paket-checkinstall] Debian-Paket *checkinstall*, http://packages.debian.org/de/stable/checkinstall
- [Debian-Paket-cupt] Debian-Paket *cupt*, http://packages.debian.org/de/stable/cupt
- [Debian-Paket-dctrl-tools] Debian-Paket *dctrl-tools*, http://packages.debian.org/de/stable/dctrl-tools
- [Debian-Paket-debconf] Debian-Paket *debconf*, http://packages.debian.org/de/stable/debconf
- [Debian-Paket-debconf-utils] Debian-Paket *debconf-utils*, http://packages.debian.org/de/stable/debconf-utils
- [Debian-Paket-debdelta] Debian-Paket *debdelta*, http://packages.debian.org/de/stable/debdelta
- [Debian-Paket-debfoster] Debian-Paket *debfoster*, http://packages.debian.org/de/stable/debfoster
- [Debian-Paket-debhelper] Debian-Paket *debhelper*, http://packages.debian.org/de/stable/debhelper
- [Debian-Paket-debian-archive-keyring] Debian-Paket *debian-archive-keyring*, http://packages.debian.org/de/stable/debian-archive-keyring
- [Debian-Paket-debian-goodies] Debian-Paket *debian-goodies*, http://packages.debian.org/de/stable/debian-goodies
- [Debian-Paket-debian-handbook] Debian-Paket *debian-handbook*, http://packages.debian.org/de/stable/debian-handbook
- [Debian-Paket-debian-security-support] Debian-Paket *debian-security-support*, https://packages.debian.org/wheezy-backports/-debian-security-support
- [Debian-Paket-debmirror] Debian-Paket *debmirror*, https://packages.debian.org/de/stable/debmirror
- [Debian-Paket-deborphan] Debian-Paket *deborphan*, http://packages.debian.org/de/stable/deborphan
- [Debian-Paket-debpartial-mirror] Debian-Paket *debpartial-mirror*, https://packages.debian.org/de/stable/debpartial-mirror
- [Debian-Paket-debtags] Debian-Paket *debtags*, http://packages.debian.org/de/stable/debtags

- [Debian-Paket-debtags-edit] Debian-Paket *debtags-edit*, http://packages.debian.org/de/stable/debtags-edit
- [Debian-Paket-debtorrent] Debian-Paket *debtorrent*, http://packages.debian.org/sid/debtorrent
- [Debian-Paket-devscripts] Debian-Paket *devscripts*, http://packages.debian.org/de/stable/devscripts
- [Debian-Paket-debsums] Debian-Paket *debsums*, http://packages.debian.org/de/stable/debsums
- [Debian-Paket-debtree] Debian-Paket *debtree*, http://packages.debian.org/de/stable/debtree
- [Debian-Paket-dgit] Debian-Paket *dgit*, https://packages.debian.org/testing/dgit
- [Debian-Paket-dh-make-perl] Debian-paket *dh-make-perl*, https://packages.debian.org/jessie/dh-make-perl
- [Debian-Paket-dkms] Debian-Paket *dkms* (Dynamic Kernel Modules Support), http://packages.debian.org/de/stable/dkms
- [Debian-Paket-dlocate] Debian-Paket *dlocate*, https://packages.debian.org/de/stable/dlocate
- [Debian-Paket-dpkg] Debian-Paket *dpkg*, https://packages.debian.org/de/stable/dpkg
- [Debian-Paket-dpkg-dev] Debian-Paket *dpkg-dev*, https://packages.debian.org/de/stable/dpkg-dev
- [Debian-Paket-dpkg-www] Debian-Paket *dpkg-www*, https://packages.debian.org/de/stable/dpkg-www
- [Debian-Paket-dwm] Debian-Paket *dwm*, http://packages.debian.org/de/stable/dwm
- [Debian-Paket-etckeeper] Debian-Paket *etckeeper*, http://packages.debian.org/jessie/etckeeper
- [Debian-Paket-galternatives] Debian-Paket *galternatives*, http://packages.debian.org/de/stable/galternatives
- [Debian-Paket-gawk] Debian-Paket *gawk*, http://packages.debian.org/de/stable/gawk
- [Debian-Paket-gcc] Debian-Paket *gcc*, http://packages.debian.org/de/stable/gcc
- [Debian-Paket-gdebi] Debian-Paket *gdebi*, http://packages.debian.org/de/stable/gdebi
- [Debian-Paket-gdebi-core] Debian-Paket *gdebi-core*, http://packages.debian.org/de/stable/gdebi-core
- [Debian-Paket-gdebi-kde] Debian-Paket *gdebi-kde*, http://packages.debian.org/de/stable/gdebi-kde
- [Debian-Paket-geoip-database] Debian-Paket *geoip-database*, https://packages.debian.org/de/stable/geoip-database
- [Debian-Paket-git-dpm] Debian-Paket *git-dpm*, https://packages.debian.org/de/stable/git-dpm
- [Debian-Paket-gnome-packagekit] Debian-Paket *gnome-packagekit*, https://packages.debian.org/jessie/gnome-packagekit
- [Debian-Paket-goplay] Debian-Paket *goplay*, https://packages.debian.org/de/stable/goplay
- [Debian-Paket-gui-apt-key] Debian-Paket *gui-apt-key*, https://packages.debian.org/de/stable/gui-apt-key
- [Debian-Paket-how-can-i-help] Debian-Paket *how-can-i-help*, https://packages.debian.org/jessie/how-can-i-help
- [Debian-Paket-ia32-libs] Debian-Paket *ia32-libs*, https://packages.debian.org/de/stable/ia32-libs
- [Debian-Paket-init] Debian-Paket *init*, http://packages.debian.org/de/stable/init
- [Debian-Paket-isenkram] Debian-Paket *isenkram*, https://packages.debian.org/de/stable/isenkram
- [Debian-Paket-isenkram-cli] Debian-Paket *isenkram-cli*, https://packages.debian.org/de/stable/isenkram-cli
- [Debian-Paket-libapache2-mod-authn-yubikey] Debian-Paket *libapache2-mod-authn-yubikey*, http://packages.debian.org/de/-stable/libapache2-mod-authn-yubikey
- [Debian-Paket-libapt-inst] Debian-Paket *libapt-inst*, http://packages.debian.org/de/stable/libapt-inst
- [Debian-Paket-libapt-pkg4.12] Debian-Paket *libapt-pkg4.12*, http://packages.debian.org/de/stable/libapt-pkg4.12
- [Debian-Paket-libapt-pkg-doc] Debian-Paket *libapt-pkg-doc*, http://packages.debian.org/de/stable/libapt-pkg-doc

- [Debian-Paket-libapt-pkg-perl] Debian-Paket *libapt-pkg-perl*, http://packages.debian.org/de/stable/libapt-pkg-perl
- [Debian-Paket-lintian] Debian-Paket *lintian*, http://packages.debian.org/de/stable/lintian
- [Debian-Paket-localepurge] Debian-Paket *localepurge*, http://packages.debian.org/stable/localepurge
- [Debian-Paket-lsb] Debian-Paket *lsb*, http://packages.debian.org/stable/lsb
- [Debian-Paket-make] Debian-Paket *make*, http://packages.debian.org/de/stable/make
- [Debian-Paket-module-assistant] Debian-Paket *module-assistant*, http://packages.debian.org/de/stable/module-assistant
- [Debian-Paket-muon] Debian-Paket *muon*, http://packages.debian.org/de/stretch/muon
- [Debian-Paket-netselect] Debian-Paket *netselect*, http://packages.debian.org/de/stable/netselect
- [Debian-Paket-netselect-apt] Debian-Paket *netselect-apt*, http://packages.debian.org/de/stable/netselect-apt
- [Debian-Paket-packagekit] Debian-Paket *packagekit*, http://packages.debian.org/de/stable/packagekit
- [Debian-Paket-packagekit-backend-aptcc] Debian-Paket *packagekit-backend-aptcc*, http://packages.debian.org/de/wheezy/packagekit-backend-aptcc
- [Debian-Paket-packagekit-backend-smart] Debian-Paket *packagekit-backend-smart*, http://packages.debian.org/de/wheezy/packagekit-backend-smart
- [Debian-Paket-packagesearch] Debian-Paket *packagesearch*, http://packages.debian.org/de/stable/packagesearch
- [Debian-Paket-perl] Debian-Paket *perl*, https://packages.debian.org/jessie/perl
- [Debian-Paket-piuparts] Debian-Paket *piuparts*, http://packages.debian.org/de/stable/piuparts
- [Debian-Paket-python-software-properties] Debian-Paket *python-software-properties*, http://packages.debian.org/de/stable/python-software-properties
- [Debian-Paket-reportbug] Debian-Paket *reportbug*, https://packages.debian.org/de/stable/reportbug
- [Debian-Paket-reprepro] Debian-Paket *reprepro*, https://packages.debian.org/de/stable/reprepro
- [Debian-Paket-rpm] Debian-Paket *rpm*, https://packages.debian.org/de/stable/rpm
- [Debian-Paket-sensible-utils] Debian-Paket *sensible-utils*, http://packages.debian.org/de/stable/sensible-utils
- [Debian-Paket-smartpm] Debian-Paket *smartpm*, http://packages.debian.org/de/stable/smartpm
- [Debian-Paket-software-center] Debian-Paket *software-center*, http://packages.debian.org/de/wheezy/software-center
- [Debian-Paket-software-properties-common] Debian-Paket *software-properties-common*, https://packages.debian.org/jessie/-software-properties-common
- [Debian-Paket-synaptic] Debian-Paket *synaptic*, http://packages.debian.org/de/stable/synaptic
- [Debian-Paket-tasksel] Debian-Paket *tasksel*, http://packages.debian.org/de/stable/tasksel
- [Debian-Paket-tzdata] Debian-Paket *tzdat*, http://packages.debian.org/de/stable/tzdata
- [Debian-Paket-util-linux] Debian-Paket *util-linux*, http://packages.debian.org/de/stable/util-linux
- [Debian-Paket-vrms] Debian-Paket *vrms*, https://packages.debian.org/stable/vrms
- [Debian-Paket-wajig] Debian-Paket *wajig*, http://packages.debian.org/de/stable/wajig
- [Debian-Paket-wget] Debian-Paket *wget*, http://packages.debian.org/de/stable/wget
- [Debian-Paket-whatmaps] Debian-Paket *whatmaps*, http://packages.debian.org/de/jessie/whatmaps
- [Debian-Paket-xara-gtk] Debian-Paket *xara-gtk*, http://packages.debian.org/de/stable/xara-gtk

- [Debian-Paket-yum] Debian-Paket *yum*, https://packages.debian.org/de/stable/yum
- [Debian-Paket-zutils] Debian-Paket *zutils*, http://packages.debian.org/de/stable/zutils
- [Debian-Policy-Manual] Debian Policy Manual, http://www.debian.org/doc/debian-policy/
- [Debian-Policy-Subsections] Debian Policy Manual, Bereich Subsections, http://www.debian.org/doc/debian-policy/ch-archive.html#subsections
- [Debian-Popcon-Graph] Debian Popcon Graphen, https://qa.debian.org/popcon-graph.php
- [Debian-Popularity-Contest] Debian Popularity Contest, http://popcon.debian.org/
- [Debian-Ports-Projekt] Debian-Ports Projekt, http://www.ports.debian.org/
- [Debian-Pure-Blends] Andreas Tille, Ben Armstrong, Emmanouil Kiagias: Debian Pure Blends, http://blends.debian.org/-blends/
- [DebianQA] Debian Quality Assurance (QA) Team, https://qa.debian.org/
- [Debian-Redirector] The Debian Redirector, http://httpredir.debian.org/
- [Debian-Security] Debian-Sicherheitsinformationen, https://www.debian.org/security/
- [Debian-Snapshots] Debian Snapshots, http://snapshot.debian.org/
- [Debian-Sources-List-Generator] Debian Sources List Generator, http://debgen.simplylinux.ch/
- [Debian-Spiegel-Informationen] Spiegel-Informationen einreichen, http://www.debian.org/mirror/submit
- [Debian-Spiegel-Liste] Liste der Debian-Mirror, http://www.debian.org/mirror/list
- [Debian-udeb] Debian-Dokumentation zu *udeb*, http://d-i.alioth.debian.org/doc/internals/ch03.html
- [Debian-Release-Notes] Veröffentlichungshinweise zur Debian-Distribution, https://www.debian.org/releases/stable/releasenotes
- [Debian-Social-Contract] Debian-Gesellschaftsvertrag, http://www.debian.org/social_contract.de.html
- [Debian-Virtual-Packages-List] Liste aller offiziell verwendeten virtuellen Pakete, http://www.debian.org/doc/packaging-manuals/-virtual-package-names-list.adoc
- [Debian-Webseite] Webseite des Debian-Projekts, http://www.debian.org/
- [Debian-Wiki-Alternatives] Debian Wiki: Debian Alternatives, https://wiki.debian.org/DebianAlternatives
- [Debian-Wiki-AptConf] Debian Wiki: Eintrag zu AptConf, https://wiki.debian.org/AptConf
- [Debian-Wiki-ARM-EABI-Port] Debian Wiki: ARM EABI Port, http://wiki.debian.org/ArmPorts
- [Debian-Wiki-chroot] Debian Wiki: `chroot` (deutschsprachig), http://wiki.debian.org/de/chroot
- [Debian-Wiki-cupt] Debian Wiki: Eintrag zu `cupt`, https://wiki.debian.org/Cupt
- [Debian-Wiki-Debian-Entwickler] Debian Wiki: Wie werde ich ein Debian-Entwickler?, http://wiki.debian.org/DebianDeveloper
- [Debian-Wiki-Maintainer] Debian Wiki: Debian Maintainer, https://wiki.debian.org/DebianMaintainer
- [Debian-Wiki-FHS] Debian Wiki: Filesystem Hierarchy Standard (FHS), http://wiki.debian.org/FilesystemHierarchyStandard
- [Debian-Wiki-Debian-GNUHurd] Debian Wiki: Debian GNU/Hurd, https://wiki.debian.org/Debian_GNU/Hurd
- [Debian-Wiki-Debian-GNUkFreeBSD] Debian Wiki: Debian GNU/kFreeBSD, https://wiki.debian.org/Debian_GNU/kFreeBSD
- [Debian-Wiki-Debian-Repository-Format] Debian Wiki: Debian Repository Format, https://wiki.debian.org/RepositoryFormat
- [Debian-Wiki-DebTorrent] Debian Wiki: DebTorrent, https://wiki.debian.org/DebTorrent
- [Debian-Wiki-DiskImage] Debian Wiki: Diskimage, https://wiki.debian.org/DiskImage

- [Debian-Wiki-FAI] Debian Wiki: FAI (Fully Automatic Installation) for Debian GNU/Linux, https://wiki.debian.org/FAI
- [Debian-Wiki-git-dpm] Debian Wiki: `git-dpm` — debian packages in git manager, http://git-dpm.alioth.debian.org/
- [Debian-Wiki-git-dpm-packaging] Debian Wiki: Maintaining Debian source packages in git with git-dpm, https://wiki.debian.org/-PackagingWithGit/GitDpm
- [Debian-Wiki-how-can-i-help] Debian Wiki: How Can I Help?, https://wiki.debian.org/how-can-i-help
- [Debian-Wiki-multiarch] Debian Wiki: Debian multiarch support, https://wiki.debian.org/Multiarch
- [Debian-Wiki-SecureApt] Debian Wiki: SecureApt, https://wiki.debian.org/SecureApt
- [Debian-Wiki-Skype] Debian Wiki: Skype, https://wiki.debian.org/skype
- [Debian-Wiki-WNPP] Debian Wiki: Work-Needing and Prospective Packages (WNPP), https://wiki.debian.org/WNPP
- [debtorrent-Projektseite] Webseite zum DebTorrent-Projekt, http://debtorrent.alioth.debian.org/
- [debtree-Projektseite] Webseite zum debtree-Projekt, http://collab-maint.alioth.debian.org/debtree/
- [DEP-8] Debian Enhancement Proposal *DEP 8*: automatic as-installed package testing, http://dep.debian.net/deps/dep8/
- [DFSG] Debian Free Software Guidelines (DFSG), https://www.debian.org/social_contract#guidelines
- [dinstall-status] dinstall Status, https://ftp-master.debian.org/dinstall.status
- [DNF-Dokumentation] Dokumentation zu Dandified YUM (DNF), http://dnf.readthedocs.org/en/latest/
- [dpkg-Kumar] Avishek Kumar: 15 Practical Examples of "dpkg commands" for Debian Based Distros, http://www.tecmint.com/-dpkg-command-examples/
- [dpmb-github] Debian Package Management Book, GitHub-Repository, https://github.com/dpmb
- [Drilling-APT-Pinning-LinuxUser] Thomas Drilling: Festgenagelt. Tricks zum Mischen von Debian-Releases, LinuxUser 06/2012, LinuxNewMedia AG, München, 2012, S. 35ff., http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2012/-06/Tricks-zum-Mischen-von-Debian-Releases
- [Drilling-Checkinstall-LinuxUser] Thomas Drilling: Gut geschnürt. Paketbau in Eigenregie mit Checkinstall, LinuxUser 06/2012, LinuxNewMedia AG, München, 2012, S. 38ff., http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2012/-06/Paketbau-in-Eigenregie-mit-Checkinstall
- [DysonOS] Dyson OS, http://osdyson.org/
- [Edubuntu] Edubuntu, http://www.edubuntu.org/
- [Emdebian] Debian für Embedded Devices, http://www.emdebian.org/
- [FHS-Linux-Foundation] Filesystem Hierarchy Standard (FHS), Linux Foundation, https://wiki.linuxfoundation.org/en/FHS
- [Finkproject] Fink-Projekt, http://www.finkproject.org/
- [Foster-Johnson-RPM-Guide] Eric Foster-Johnson, Stuart Ellis und Ben Cotton: RPM Guide, 2005/2011, Fedora Project Contributors, Edition 0, http://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/index.html
- [FreeBSD] FreeBSD-Projekt, http://www.freebsd.org/
- [FreeCode] FreeCode, http://freecode.com/
- [gambaru-rc-alert] gambaru.de: Wie man veröffentlichungskritische Bugs in Debian beseitigt, http://www.gambaru.de/blog/-2012/09/19/wie-man-veroffentlichungskritische-bugs-in-debian-beseitigt/
- [gdebi] Gdebi, https://launchpad.net/gdebi
- [geoiptool] Geo IP Tool, http://www.geoiptool.com/
- [GitHub] GitHub, https://github.com/

- [github-issue] Issue auf GitHub, https://github.com/dpmb/dpmb/issues/new
- [github-pull-request] Pull-Request mitsamt Patch auf GitHub, https://github.com/dpmb/dpmb/compare
- [GNU-Linux-Distribution-Timeline] GNU Linux Distribution Timeline, http://futurist.se/gldt
- [GObject-Introspection] GObject Introspection Middleware, https://wiki.gnome.org/Projects/GObjectIntrospection
- [Graphviz] Graphviz — Graph Visualization Software, http://www.graphviz.org/
- [Grml] Grml, http://www.grml.org/
- [Gtkorphan] Gtkorphan, Webseite zum Programm, http://www.marzocca.net/linux/gtkorphan.html
- [Hackerfunk] Hackerfunk Zürich, Folge 65, Fachliteratur Schreiben, http://www.hackerfunk.ch/?id=127
- [Heinlein-LPIC-1] Peer Heinlein: LPIC-1. Vorbereitung auf die Prüfung des Linux Professional Institute, OpenSource Press, 5. Auflage, ISBN 978-3-95539-012-9, deutsch, 501 Seiten, http://www.opensourcepress.de/de/produkte/LPIC-1/452/978-3-95539-012-9
- [Hertzog-Mas-Debian-Administrators-Handbook] Raphael Hertzog, Roland Mas: The Debian Administrator's Handbook, 2012, ISBN 979-10-91414-00-5, http://debian-handbook.info/
- [Hertzog-Obsolete-Packages] Raphael Hertzog: Debian Cleanup Tip #2: Get rid of obsolete packages, http://raphaelhertzog.com/-2011/02/07/debian-cleanup-tip-2-get-rid-of-obsolete-packages/
- [Hofmann-Debtags-LinuxUser] Frank Hofmann: Dschungelführer. Pakete zielgenau finden mit Debtags, LinuxUser 06/2012, LinuxNewMedia AG, München, 2012, S. 22ff., http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2012/-06/Pakete-zielgenau-finden-mit-Debtags
- [Hofmann-Debtags-Vortrag] Frank Hofmann: Debian-Pakete zielgenau finden mit Debtags, Vortrag im Rahmen des Linuxday Dornbirn, November 2013, http://fileshare.lugv.at/public/vortraege2013/linuxday/debian-debtags.pdf
- [Hofmann-Osterried-Alien-LinuxUser] Frank Hofmann, Thomas Osterried: Gestaltwandler. Programmpakete richtig konvertieren, LinuxUser 1/2010, LinuxNewMedia AG, München, 2010, S. 32ff., http://www.linux-community.de/Internal/Artikel/-Print-Artikel/LinuxUser/2010/01/Programmpakete-richtig-konvertieren
- [Hofmann-Smartpm-LinuxUser] Frank Hofmann: Mit allen Extras. Debian-Pakete verwalten mit dem Smart Package Manager, LinuxUser 07/2013, LinuxNewMedia AG, München, 2013, S. 68ff., http://www.linux-community.de/Internal/Artikel/Print-Artikel/LinuxUser/2013/07/Debian-Pakete-verwalten-mit-dem-Smart-Package-Manager
- [Hofmann-Webseite] Frank Hofmanns Webseite, http://www.efho.de/
- [Hofmann-Winde-Aptsh-LinuxUser] Frank Hofmann, Thomas Winde: Zentraler Zugangspunkt. Komfortabel Pakete managen mit der Apt-Shell, LinuxUser 06/2012, LinuxNewMedia AG, München, 2012, S. 30ff., http://www.linux-community.de/-Internal/Artikel/Print-Artikel/LinuxUser/2012/06/Komfortabel-Pakete-managen-mit-der-Apt-Shell
- [Hurd] GNU Hurd Projekt, http://www.gnu.org/software/hurd/
- [Huy-Tran-Apt-Mirror] Huy Tran: How to update and upgrade with fastest mirror from the command line, http://www.namhuy.net/-1040/how-to-update-and-upgrade-with-fastest-mirror-from-the-command-line.html
- [Illumian] Illumian, http://illumian.org/
- [ipbrick] IPBRICK, http://www.ipbrick.de/
- [ipkg] Itsy Package Management System (IPKG) bei Wikipedia, http://de.wikipedia.org/wiki/IPKG
- [Isenkram-Reinholdtsen] Isenkram im Blog von Petter Reinholdtsen, http://people.skolelinux.org/pere/blog/tags/isenkram/
- [Java-Apt] Java Annotation Processing Tool, https://metro.java.net/1.5/docs/apt.html
- [Jurzik-Debian-Handbuch] Heike Jurzik: Debian GNU/Linux: Das umfassende Handbuch, Verlag: Galileo Computing; 5. Auflage, 2013, ISBN-13: 978-3-8362-2661-5
- [Kali-Linux] Kali Linux, http://www.kali.org/

- [Kemp-dh-make-perl] Steve Kemp: Building Debian packages of Perl modules, https://www.debian-administration.org/article/-78/Building_Debian_packages_of_Perl_modules
- [Kemp-dget] Steve Kemp: Downloading Debian source packages easily, http://www.debian-administration.org/article/504/-Downloading_Debian_source_packages_easily
- [Keryx] Keryx im Ubuntu Launchpad, https://launchpad.net/keryx
- [Knoppix] Knoppix, http://www.knopper.net/knoppix/
- [Kofler-Linux-2013] Michael Kofler: Linux 2013. Das Desktop- und Server-Handbuch für Ubuntu, Debian, CentOS und Co. (Open Source Library), Addison-Wesley Verlag, 2013, ISBN 978-3827332080, S. 480-490, S. 1112-1115
- [Krafft-Debian-System] Martin F. Krafft: Das Debian-System. Konzepte und Methoden, Open Source Press München, 2006, deutsche Ausgabe, Erstauflage, S. 140 f.
- [Krafft-Debian-System144] Ebd., S. 144 ff.
- [Krafft-Debian-System137ff] Ebd., Kapitel 5, S. 137-294
- [LernStick] LernStick, Fachhochschule Nordwestschweiz, Solothurn, http://www.imedias.ch/projekte/lernstick/index.cfm
- [libelektra] Libelektra, http://community.libelektra.org/wp/?p=145
- [LiMux] LiMux — Linux in der Stadtverwaltung München, http://www.muenchen.de/rathaus/Stadtverwaltung/Direktorium/-LiMux.html
- [Lintian] Lintian-Projekt, https://lintian.debian.org/
- [LinuxMint] Linux Mint, http://www.linuxmint.com/
- [localepurge] *localepurge*, Projektseite im Linux Wiki, http://linuxwiki.de/localepurge
- [Loschwitz-Sourceformat] Martin Loschwitz: Zusammenpacken! Das neue Sourceformat für Debian-Pakete, Linux-Magazin 06/2011, http://www.linux-magazin.de/Ausgaben/2011/06/Debian-Src-3.0
- [lpic-101] Linux Professional Institute, Unterlagen für LPIC 101, https://www.lpi.org/study-resources/lpic-1-101-exam-objectives/-
- [lug.berlin] Das Berliner Community-Portal lug.berlin, http://lug.berlin/
- [Maemo] Maemo Community, http://maemo.org/
- [Mageia-urpmi] `urpmi` — Werkzeuge zur Paketverwaltung bei Mageia, Mageia Wiki, https://wiki.mageia.org/de/URPMI
- [Mandriva-Wiki] Mandriva Control Center im Mandriva Wiki, http://wiki.mandriva.com/en/Tools/Control_Center
- [Maassen-LPIC-1] Harald Maaßen: LPIC-1. Sicher zur erfolgreichen Linux-Zertifizierung, Rheinwerk Computing, Bonn, 4. Auflage, 2015, ISBN 978-3-8362-3527-3, https://www.rheinwerk-verlag.de/lpic-1_3781/
- [MeeGo] MeeGo, https://meego.com/
- [mime-applications-associations] MIME Application Associations, http://www.freedesktop.org/wiki/Specifications/mime-apps-spec/
- [mime-applications-associations-default-applications] Default Applications, http://standards.freedesktop.org/mime-apps-spec/-latest/ar01s04.html
- [Naumann-Abakus-Internet] Dr. Friedrich Naumann: Vom Abakus zum Internet: die Geschichte der Informatik. Darmstadt, Primus-Verlag, 2001, ISBN 3-89678-224-X
- [Ncurses] Ncurses-Projektseite beim GNU-Projekt, http://www.gnu.org/software/ncurses/
- [Neo900] Neo900-Projekt, http://neo900.org/
- [NexentaOS] Wikipedia-Eintrag zu Nexenta OS, http://en.wikipedia.org/wiki/Nexenta_OS

- [nixcraft-apt-get] `apt-get`-Spickzettel im Nixcraft-Blog, http://www.cyberciti.biz/howto/question/linux/apt-get-cheat-sheet.php
- [nixcraft-blog] Nixcraft-Blog, http://www.cyberciti.biz/tips/linux-debian-package-management-cheat-sheet.html
- [nixcraft-dpkg] `dpkg`-Spickzettel im Nixcraft-Blog, http://www.cyberciti.biz/howto/question/linux/dpkg-cheat-sheet.php
- [OpenMoko] OpenMoko-Projekt, http://www.openmoko.org/
- [opkg] OpenMoko Package Format, http://wiki.openmoko.org/wiki/Opkg
- [PackageKit] Webseite zu PackageKit, http://www.packagekit.org/
- [Pacman-Rosetta] Pacman Rosetta — Vergleich der Kommandozeilenparameter von `pacman`, `yum`, `apt-get`, `rug`, `zypper` und `emerge`, ArchLinux-Wiki, https://wiki.archlinux.org/index.php/Pacman_Rosetta
- [Piuparts] Piuparts (Package Installation, UPgrading And Removal Testing Suite), https://piuparts.debian.org/
- [Plura-lts] Michael Plura: Am Leben halten, ix 12/2014, http://www.heise.de/ix/heft/Am-Leben-halten-2458886.html
- [RaspberryPi] Webseite zur Hardwareplattform Raspberry Pi, http://www.raspberrypi.org/
- [Raspbian] Debian für das Raspberry Pi, http://www.raspbian.org/
- [RFC822] RFC 822: Standard For The Format Of Text Messages, IETF, https://www.ietf.org/rfc/rfc0822.adoc
- [Ritesh-apt-offline] Ritesh Sarraf: Offline Package Management for APT, https://www.debian-administration.org/article/648/-Offline_Package_Management_for_APT
- [RM-software-center] Entfernung von Ubuntu Software Center aus Debian, https://bugs.debian.org/755452
- [RMLL] Rencontres Mondiales du Logiciel Libre, http://rmll.info/
- [RPM-Canepa] Gabriel Cánepa: Linux Package Management with Yum, RPM, Apt, Dpkg, Aptitude and Zypper – Part 9, http://www.tecmint.com/linux-package-management/
- [RPM-Gite] Vivek Gite: CentOS / RHEL: See Detailed History Of yum Commands, http://www.cyberciti.biz/faq/yum-history-command/
- [RPM-Salve] Ravi Salve: 20 Practical Examples of RPM Commands in Linux, http://www.tecmint.com/20-practical-examples-of-rpm-commands-in-linux/
- [rpmseek] Rpmseek, http://www.rpmseek.com/
- [RPM-Webseite] Dokumentation auf rpm.org, http://www.rpm.org/wiki/Docs
- [RPM-Verify] When Verification Fails — rpm -V Output, http://www.rpm.org/max-rpm/s1-rpm-verify-output.html
- [Schnober-Checkinstall-LinuxUser] Carsten Schnober: Wie am Schnürchen. Debian-Pakete bauen von einfach bis anspruchsvoll, LinuxUser 02/2008, LinuxNewMedia AG, München, 2008, S. 88ff., https://www.linux-user.de/ausgabe/2008/02/088/-index.html
- [screenshots.debian.net] Screenshot-Sammlung von Debian- und Ubuntu-Paketen, https://screenshots.debian.net/
- [Sentinel4Mobile] Sentinel4Mobile Berlin, Werner Heuser, http://sentinel4mobile.de/
- [Siduction] Siduction, http://siduction.org/
- [SingleClickInstall] https://wiki.ubuntu.com/SingleClickInstall
- [Skolelinux] Skolelinux, http://skolelinux.de/
- [Skype] Skype, www.skype.com/
- [SmartPM] Smart Package Manager, Projektseite, http://labix.org/smart
- [SourceForge] SourceForge, https://sourceforge.net/

- [Stackexchange-LTS] How to work around „Release file expired“ problem on a local mirror, http://unix.stackexchange.com/-questions/2544/how-to-work-around-release-file-expired-problem-on-a-local-mirror
- [Stapelberg-Debian-Repo] Michael Stapelberg: Kurz-Howto: Eigenes Debian-Repository aufbauen, http://michael.stapelberg.de/-Artikel/Debian_Repository/
- [SteamOS] Steam OS, http://store.steampowered.com/steamos/
- [StormOS] StormOS, Wiki-Seite im Debian Derivative Census, http://wiki.debian.org/Derivatives/Census/StormOS
- [Suter-apt-offline] Samuel Suter: apt offline benutzen, http://www.lugs.ch/lib/doc/apt-offline.phtml
- [SWITCH] SWITCH, das Hochleistungsnetzwerk der Schweizer Hochschulen, http://www.switch.ch/
- [Tanglu] Tanglu GNU/Linux, http://www.tanglu.org/de/
- [ToyStory] Toy Story im Disney Wiki, http://disney.wikia.com/wiki/Toy_Story
- [Ubuntu] Ubuntu Linux, http://www.ubuntu.com/
- [Ubuntu-apturl] AptURL im Ubuntu Apps Directory, https://apps.ubuntu.com/cat/applications/apturl/
- [Ubuntu-Paket-software-center] Ubuntu-Paket *software-center*, https://launchpad.net/software-center
- [Ubuntu-Paket-ubuntu-keyring] Ubuntu-Paket *ubuntu-keyring*, http://packages.ubuntu.com/de/trusty/ubuntu-keyring
- [Ubuntu-Landscape] Ubuntu Landscape System Management, https://landscape.canonical.com/
- [Ubuntu-Launchpad] Ubuntu Launchpad, https://launchpad.net/ubuntu
- [Ubuntu-One] Ubuntu One, http://ubuntuone.com
- [Ubuntu-One-Wikipedia] Ubuntu One, Wikipedia-Eintrag, http://de.wikipedia.org/wiki/Ubuntu_One
- [Ubuntu-Snappy] Ubuntu Package Format Snappy, https://developer.ubuntu.com/en/snappy/
- [Ubuntu-Snappy-Projekt] Ubuntu Package Format Snappy (Projektseite), https://snapcraft.io/
- [Ubuntu-Software-Center] Ubuntu Software Center, Projektseite/Wiki, https://wiki.ubuntu.com/SoftwareCenter
- [Ubuntu-Sources-List-Generator] Ubuntu Sources List Generator, http://repogen.simplylinux.ch/
- [Ultimate-Debian-Database] Ultimate Debian Database, https://udd.debian.org/
- [UCS] Univention Corporate Server (UCS), http://www.univention.de/produkte/ucs/
- [univention-errata] Aktualisierungen bei UCS, https://errata.univention.de/
- [Vogt-apturl] Michael Vogt: apturl bei Ubuntu Users, http://wiki.ubuntuusers.de/apturl
- [Vogt-Apt-1.0] Michael Vogt: apt 1.0, http://mvogt.wordpress.com/2014/04/04/apt-1-0/
- [Vogt-Apt-Mirror] Michael Vogt: The apt mirror method, https://mvogt.wordpress.com/2011/03/21/the-apt-mirror-method/
- [Vogt-gdebi] Michael Vogt: Using gdebi to install build-dependencies, http://mvogt.wordpress.com/2013/03/22/using-gdebi-to-install-build-dependencies/
- [wajig-Webseite] Webseite des wajig-Projekts, http://wajig.togaware.com/
- [Watson-App-Design] Colin Watson: App installer design: click packages, https://lists.ubuntu.com/archives/ubuntu-devel/-2013-May/037074.html
- [Wheezy-Paketliste] Paketliste zu Debian *Wheezy*, http://packages.debian.org/wheezy/
- [Wizards-of-Foss] Wizards of FOSS, Berlin, http://wizards-of-foss.de/
- [Wizards-of-Foss-Blog] Blog der Wizards of FOSS, http://wizards-of-foss.de/de/weblog/

- [xfce] XFCE Window Manager, http://www.xfce.org/
- [xtronics-Wiki] Wiki bei xtronics, http://wiki.xtronics.com/index.php/Wajig
- [xubuntu-apt-offline] xubuntu Offline Documentation, http://docs.xubuntu.org/1304/offline-packages.html
- [YUM] Yellowdog Updater, Modified (YUM), Projektseite, http://yum.baseurl.org/
- [YUM-Salve] Ravi Salve: 20 Linux YUM (Yellowdog Updater, Modified) Commands for Package Management, http://www.tecmint.c
  20-linux-yum-yellowdog-updater-modified-commands-for-package-mangement/
- [Zypper] Zypper, Projektseite, http://de.opensuse.org/Zypper

Teil V

Anhang

Kapitel 45

Debian-Architekturen

45.1 Offizielle Architekturen

Debian unterstützt die folgenden Architekturen und Plattformen:

i386 (i486/i586)
x86 (PC) 32-Bit. Trotz des Namens ab Debian 8 *Jessie* nur Pentium-kompatible Hardware. In Debian 6 *Squeeze* und 7 *Wheezy* umfasst das noch i486, jedoch bereits keine i486 SX mehr.

amd64 (x86_64)
x86 (PC) 64-Bit. AMD 64-Bit-CPUs mit AMD64-Erweiterung und Intel CPUs mit EM64T-Erweiterung.

arm64 (aarch64-linux-gnu)
ARM 64 Bit. 64-Bit-ARMv8-Architektur, die im Vergleich zu anderen ARM-Architekturen einen aufgeräumten Befehlssatz hat und auf den Servermarkt zielt. Hardware-Architektur des iPhone 5.

armel (arm-linux-gnueabi)
ARM (EABI) [Debian-Wiki-ARM-EABI-Port].

armhf (arm-linux-gnueabihf)
ARM. Hardware Floating Point ABI, ab Debian 7.0 *Wheezy* unterstützt [Debian-Wiki-ARM-EABI-Port].

hurd-i386 (i486-gnu)
32-Bit-PC (i386) mit GNU-Hurd-Kernel (nur in Debian *unstable*).

mips (mips-linux-gnu)
MIPS (Big Endian).

mipsel (mipsel-linux-gnu)
MIPS (Little Endian).

powerpc (powerpc-linux-gnu)
Motorola/IBM PowerPC. Läuft auf vielen der Apple Macintosh PowerMac-Modelle sowie auf Rechnern der offenen CHRP- und PReP-Architekturen.

ppc64el (powerpc64le-linux-gnu)
PowerPC 64-Bit (POWER7+, POWER8). Little-Endian-Portierung von ppc64, nutzt die neue OpenPower-ELFv2-ABI.

s390x (s390x-linux-gnu)
IBM S/390 und zSeries, 64-Bit-Userland.

45.2 Veraltete Architekturen

Folgende Architekturen werden mit Debian 8.0 *Jessie* nicht mehr unterstützt:

ia64 (ia64-linux-gnu)
Intel Itanium IA-64. Wird auch in *unstable* nicht mehr unterstützt.

kfreebsd-amd64 (x86_64-kfreebsd-gnu)
x86 (PC) 64-Bit mit FreeBSD-Kernel. Technologievorschau, seit Debian 6.0 *Squeeze*, nur noch in *unstable*.

kfreebsd-i386 (i486-kfreebsd-gnu/i586-kfreebsd-gnu)
x86 (PC) 32-Bit mit FreeBSD-Kernel. Technologievorschau, seit Debian 6.0 *Squeeze* nur noch in *unstable*. Seit 2014 nur Pentium-kompatible Hardware, in Debian 6 *Squeeze* und 7 *Wheezy* werden auch noch i486 unterstützt.

s390 (s390-linux-gnu)
IBM S/390 und zSeries. Wird auch in *unstable* nicht mehr unterstützt und wurde durch s390x ersetzt.

sparc (sparc-linux-gnu)
SPARC, 64-Bit-Kernel, 32-Bit-Userland. Nur noch in *unstable*, der Nachfolger *sparc64* ist bislang noch nicht fertig.

Die nachfolgenden Architekturen werden von Debian offiziell nicht mehr unterstützt, aber vom Debian-Ports-Projekt noch am Leben gehalten:

alpha (alpha-linux-gnu)
Alpha. Unterstützt von Debian 2.1 *Slink* bis Debian 6.0 *Squeeze*.

hppa (hppa-linux-gnu)
HP PA-RISC. Unterstützt von Debian 3.0 *Woody* bis Debian 6.0 *Squeeze*.

m68k (m68k-linux-gnu)
Motorola 68k. Die Unterstützung war komplett eingestellt, wurde aber wiederbelebt.

Diese Architekturen sind gar nicht mehr verfügbar:

arm (arm-linux-gnu)
ARM (OABI). Seit Debian 6.0 *Squeeze* nicht mehr unterstützt und durch *armel* ersetzt.

m32
M32R. Portierung auf die 32-Bit-RISC-Mikroprozessoren von Renesas Technology.

netbsd-i386 (i486-netbsd)
32-Bit-PC (i386). Portierung auf den NetBSD-Kernel.

netbsd-alpha (alpha-netbsd)
Alpha. Portierung auf den NetBSD-Kernel.

45.3 Architekturen, deren Unterstützung vorgesehen ist

Folgende Architekturen werden vom Debian-Ports-Projekt bereits unterstützt und auf eine Aufnahme in Debian vorbereitet:

powerpcspe (powerpc-linux-gnuspe)
PowerPC-Prozessoren mit Signal Processing Engine.

ppc64 (powerpc64-linux-gnu)
64-Bit-PowerPC-Prozessoren mit VMX-Einheit.

sh4
SuperH, eine Portierung auf Hitachis SuperH-Prozessoren.

sparc64 (sparc64-linux-gnu)
SPARC, vollständig 64-Bit.

x32 (x86_64-linux-gnux32)
x86 (PC) 32-Bit auf AMD 64-Bit-CPUs mit AMD64-Erweiterung und Intel CPUs mit EM64T-Erweiterung.

Folgende Architektur existiert und wird von einzelnen Entwicklern aufgebaut, ist aber beim Debian-Ports-Projekt noch nicht verfügbar:

avr32 (avr32-linux-gnu)
Atmel 32-Bit RISC. Portierung auf Atmels 32-Bit RISC-Architektur AVR32; scheint teilweise wieder eingeschlafen zu sein.

Kapitel 46

Kommandos zur Paketverwaltung im Vergleich

46.1 Zusammenfassung

Hier stellen wir Ihnen die jeweiligen Kommandos zur Paketverwaltung gegenüber, soweit dieser Schritt möglich ist. Im Rampenlicht stehen RPM, Yellowdog Updater Modified (YUM), seine Nachfolger Effing Package Management (FPM) und Dandified YUM (DNF), Debian's `dpkg`, APT sowie `aptitude`. Wir betrachten dabei bspw. die Installation und die Entfernung von Paketen, das Auflisten der installierten und verfügbaren Pakete sowie das Anzeigen von Paketabhängigkeiten.

Diese Zusammenstellung basiert auf umfangreichen Tests und Recherchen, bspw. mit Fedora 23, CentOS 6 und 7 sowie Ubuntu 15.10 *Wily Werewolf* und Debian 8 *Jessie*. Die Verfügbarkeit der Werkzeuge und deren einzelne Schalter variiert recht stark [RPM-Salve] [RPM-Gite] [RPM-Canepa] [YUM-Salve] [dpkg-Kumar] [apt-Salve] [DNF-Dokumentation].

46.2 Paket installieren

Mit den nachfolgenden Aufrufen installieren Sie ein Softwarepaket über die Kommandozeile. Unter „Pakete installieren“ in Abschnitt 8.36 gehen wir detaillierter auf `dpkg`, APT und `aptitude` ein. Unter „Installation zwischengespeicherter Pakete aus dem Paketcache“ in Abschnitt 8.32 erfahren Sie mehr darüber, wie Sie Pakete installieren, die bereits im Paketcache vorliegen.

Tabelle 46.1: Installation eines Pakets

Werkzeug	Aufruf	Anmerkungen
APT	`apt-get install` *Paketname*	Paket wird dem Paketrepository entnommen
	`apt-get --no-download install` *Paketname*	Paket wird dem Paketcache entnommen und nicht vom Paketmirror bezogen
`aptitude`	`aptitude install` *Paketname*	Paket wird dem Paketrepository entnommen
`DNF`	`dnf install` *Paketname*	Paket wird dem Paketrepository entnommen
	`dnf install` *Paketdatei*	Paketdatei liegt also lokale Datei vor
`dpkg`	`dpkg -i` *Paketname*	Paket liegt lokal als Datei vor
`RPM`	`rpm -i` *Paketname*	Paket liegt lokal als Datei vor
	`rpm -i --nodeps` *Paketname*	Paket liegt lokal als Datei vor, Installation ohne Berücksichtigung der Paketabhängigkeiten
	`rpm -ihv` *Paketname*	Paket liegt lokal als Datei vor, Installation mit Kommentaren
`YUM`	`yum localinstall` *Paketname*	Paket liegt lokal als Datei vor
	`yum install` *Paketname*	Paket wird dem Paketrepository entnommen

46.3 Paket aktualisieren

Mit den nachfolgenden Aufrufen aktualisieren Sie ein Softwarepaket über die Kommandozeile. Unter „Pakete aktualisieren“ in Abschnitt 8.39 gehen wir detaillierter auf die Vorgehensweise und die begrifflichen Unterschiede bei `dpkg`, APT, `apt` und `aptitude` ein.

Tabelle 46.2: Aktualisieren eines Pakets

Werkzeug	Aufruf	Anmerkungen
APT	`apt-get upgrade` *Paketname*	aktualisierte Version des Pakets einspielen
`apt`	`apt safe-upgrade` *Paketname*	aktualisierte Version des Pakets einspielen
`aptitude`	`aptitude safe-upgrade` *Paketname*	aktualisierte Version des Pakets einspielen
`DNF`	`dnf update`	alle installierten Pakete aktualisieren (veraltet)
	`dnf update` *Paketname*	nur *Paketname* aktualisieren (veraltet)
	`dnf upgrade`	alle installierten Pakete aktualisieren
	`dnf upgrade` *Paketname*	nur *Paketname* aktualisieren
`RPM`	`rpm -U` *Paketname*	Paket liegt lokal als Datei vor
	`rpm --upgrade` *Paketname*	Paket liegt lokal als Datei vor
`YUM`	`yum update`	die gesamte Veröffentlichung aktualisieren
`	`yum update` *Paketname*	nur *Paketname* aktualisieren

46.4 Paket löschen / entfernen

Mit den nachfolgenden Aufrufen entfernen Sie ein Softwarepaket über die Kommandozeile. Unter „Pakete deinstallieren“ in Abschnitt 8.41 gehen wir detaillierter auf `dpkg`, APT und `aptitude` ein.

Tabelle 46.3: Entfernen eines Pakets

Werkzeug	Aufruf	Anmerkungen
APT	`apt-get remove` *Paketname*	entfernt das Paket, die Konfigurationsdateien des Pakets bleiben erhalten
	`apt-get purge` *Paketname*	entfernt das Paket inklusive der Konfigurationsdateien des Pakets
	`apt-get --purge remove` *Paketname*	entfernt das Paket inklusive der Konfigurationsdateien des Pakets
`aptitude`	`aptitude remove` *Paketname*	entfernt das Paket, die Konfigurationsdateien des Pakets bleiben erhalten
	`aptitude purge` *Paketname*	entfernt das Paket inklusive der Konfigurationsdateien des Pakets
`DNF`	`dnf erase` *Paketname*	Paket wird deinstalliert (veraltet)
	`dnf remove` *Paketname*	Paket wird deinstalliert
`dpkg`	`dpkg -r` *Paketname*	entfernt nur das Paket
	`dpkg --remove` *Paketname*	entfernt nur das Paket
	`dpkg -P` *Paketname*	entfernt das Paket und die Konfigurationsdateien des Pakets
	`dpkg --purge` *Paketname*	entfernt das Paket und die Konfigurationsdateien des Pakets
`RPM`	`rpm -e` *Paketname*	entfernt das angegebene Paket
	`rpm --erase` *Paketname*	entfernt das angegebene Paket
	`rpm -e --nodeps` *Paketname*	Entfernung des Pakets ohne Berücksichtigung der Paketabhängigkeiten

Tabelle 46.3: (continued)

Werkzeug	Aufruf	Anmerkungen
YUM	`yum remove` *Paketname*	entfernt das angegebene Paket samt seiner Abhängigkeiten
	`yum erase` *Paketname*	entfernt nur das angegebene Paket

46.5 Alle installierten Pakete auflisten

Mit den nachfolgenden Aufrufen listen Sie die vorhandenen Softwarepakete über die Kommandozeile auf. Für `dpkg`, APT und `aptitude` besprechen wir das detaillierter unter „Liste der installierten Pakete anzeigen und deuten“ in Abschnitt 8.5 sowie unter „Aktualisierbare Pakete anzeigen“ in Abschnitt 8.12.

Tabelle 46.4: Softwarepakete auflisten

Werkzeug	Aufruf	Anmerkungen
APT	`apt-cache search` *Paketname*	
`aptitude`	`aptitude search ~i`	alle installierten Pakete auflisten
	`aptitude versions` *Paketname*	alle verfügbaren Pakete für *Paketname* auflisten, auch die (noch) nicht installierten Varianten
DNF	`dnf list installed`	alle installierten Pakete anzeigen
`dpkg`	`dpkg -l`	alle installierten Pakete auflisten
	`dpkg --list`	alle installierten Pakete auflisten
RPM	`rpm -qa`	alle installierten Pakete auflisten
	`rpm -qa --last`	alle zuletzt installierten Pakete auflisten, auch die (noch) nicht installierten Pakete
YUM	`yum list` *Paketname*	anzeigen, welche Versionen des Pakets installiert sind
	`yum list all`	alle installierten Pakete auflisten
	`yum list installed`	alle installierten Pakete auflisten

46.6 Einzelpaket auflisten

Mit den nachfolgenden Aufrufen listen Sie die Informationen bzw. den Installationsstatus zu einem einzelnen Softwarepaket auf. Unter „Liste der installierten Pakete anzeigen und deuten“ in Abschnitt 8.5 besprechen wir das zu `dpkg` und `aptitude` genauer.

Tabelle 46.5: Einzelnes Softwarepaket auflisten

Werkzeug	Aufruf	Anmerkungen
`aptitude`	`aptitude show` *Paketname*	
DNF	`dnf info` *Paketname*	Informationen zu Paketname anzeigen
	`dnf list installed`	alle installierten Pakete anzeigen
	`dnf list installed` *Paketname*	Installationsstatus zu Paketname anzeigen
`dpkg`	`dpkg -l` *Paketname*	Ausgabe des Installationsstatus
	`dpkg --list` *Paketname*	Ausgabe des Installationsstatus

Tabelle 46.5: (continued)

Werkzeug	Aufruf	Anmerkungen
	`dpkg -s` *Paketname*	Ausgabe der Paketinformationen
	`dpkg --status` *Paketname*	Ausgabe der Paketinformationen
`RPM`	`rpm -q` *Paketname*	Ausgabe des Installationsstatus für *Paketname*
	`rpm --query` *Paketname*	Ausgabe des Installationsstatus für *Paketname*
	`rpm -qp` *Paketname*	analog zu `-q`
`YUM`	`yum list` *Paketname*	anzeigen, welche Versionen des Pakets installiert sind

46.7 Abhängigkeiten anzeigen

Mit den nachfolgenden Aufrufen zeigen Sie die Abhängigkeiten zu anderen Paketen an. Für `dpkg` und APT gehen wir dazu genauer in „Paketabhängigkeiten anzeigen“ in Abschnitt 8.17 ein.

Tabelle 46.6: Paketabhängigkeiten anzeigen

Werkzeug	Aufruf	Anmerkungen
APT	`apt-cache depends` *Paketname*	
	`apt-cache rdepends` *Paketname*	umgekehrte Abhängigkeiten anzeigen
`RPM`	`rpm -R` *Paketname*	das Paket muß dazu lokal als Datei vorliegen
`	`rpm --requires` *Paketname*	das Paket muß dazu lokal als Datei vorliegen
	`rpm -qpR` *Paketname*	das Paket muß dazu lokal als Datei vorliegen
`YUM`	`yum deplist` *Paketname*	
	`yum info` *Paketname*	

46.8 Alle Dateien eines installierten Pakets anzeigen

Mit den nachfolgenden Aufrufen zeigen Sie an, welche Dateien und Verzeichnisse zu dem installierten Paket gehören. Für Debianpakete widmen wir uns dem Thema in „Paketinhalte anzeigen“ in Abschnitt 8.23.

Tabelle 46.7: Paketinhalte anzeigen

Werkzeug	Aufruf	Anmerkungen
APT		
`dpkg`	`dpkg -L` *Paketname*	
	`dpkg --listfiles` *Paketname*	
`dpkg-query`	`dpkg-query -L` *Paketname*	
	`dpkg-query --listfiles` *Paketname*	
`RPM`	`rpm -ql` *Paketname*	
`YUM`		

46.9 Paket identifizieren, aus dem eine Datei stammt

Um herauszufinden, aus welchem Paket eine Datei stammt, bieten sowohl `rpm` als auch `dpkg` entsprechende Schalter an. Für Debianpakete besprechen wir das Thema in „Paket zu Datei finden“ in Abschnitt 8.22 ausführlich.

Tabelle 46.8: Paket zu Datei finden

Werkzeug	Aufruf	Anmerkungen
APT	`apt-file search` *Dateiname*	Suche in allen verfügbaren Paketen
`aptitude`	kein Schalter	
`DNF`	`dnf provides` *Dateiname*	*Dateiname* umfaßt hier den vollständigen Namen inklusive Pfad
`dpkg`	`dpkg -S` *Dateiname*	Suche in den installierten Paketen
	`dpkg --search` *Dateiname*	Suche in den installierten Paketen
`dpkg-query`	`dpkg-query -S` *Dateiname*	Suche in den installierten Paketen
	`dpkg-query --search` *Dateiname*	Suche in den installierten Paketen
`RPM`	`rpm -qf` *Dateiname*	*Dateiname* umfaßt hier den vollständigen Namen inklusive Pfad
`YUM`	`yum provides` *Dateiname*	*Dateiname* umfaßt hier den vollständigen Namen inklusive Pfad

46.10 Paketstatus anzeigen

Diese Information zeigen Ihnen `dpkg` und `apt-cache` an. Ausführlicher beschäftigt sich damit der Abschnitt „Paketstatus erfragen“ in Abschnitt 8.4.

Tabelle 46.9: Paketstatus erfragen

Werkzeug	Aufruf	Anmerkungen
`apt`	`apt-cache show` *Paketname*	Suche in allen verfügbaren Paketen
`aptitude`	`aptitude show` *Paketname*	Suche in allen verfügbaren Paketen
`dpkg`	`dpkg -s` *Paketname*	*Paketname* muß lokal installiert sein
	`dpkg --status` *Paketname*	*Paketname* muß lokal installiert sein
	`dpkg -i` *Dateiname*	*Dateiname* bezeichnet eine lokale Datei
	`dpkg --info` *Dateiname*	*Dateiname* bezeichnet eine lokale Datei
`dpkg-query`	`dpkg-query -s` *Paketname*	*Paketname* muß lokal installiert sein
	`dpkg-query --status` *Paketname*	*Paketname* muß lokal installiert sein
`RPM`	`rpm -qi` *Paketname*	*Paketname* muß lokal installiert sein
	`rpm -qip` *Dateiname*	*Dateiname* muß lokal vorliegen
`YUM`	`yum info` *Paketname*	*Paketname* muß lokal installiert sein

46.11 Aktualisierbare Pakete anzeigen

Viele Pakete werden regelmäßig aktualisiert. Welches Kommando Ihnen die Pakete anzeigt, die in einer neuen Version bereitstehen, zeigt Ihnen die nachfolgende Tabelle. In Abschnitt ``Aktualisierbare Pakete anzeigen`` Abschnitt 8.12 erfahren Sie dazu mehr Details.

Tabelle 46.10: Aktualisierbare Pakete anzeigen

Werkzeug	Aufruf	Anmerkungen
APT	`apt-get upgrade -u`	alle Pakete auflisten, für die eine neue Version bereitsteht
	`apt-get upgrade --show-upgraded`	analog zu `-u` (Langform)
	`apt-get upgrade -u -s`	Simulation, analog zu `-u`
	`apt-get upgrade --show-upgraded --simulate`	Simulation, analog zu `-u -s` (Langform)
`aptitude`	`aptitude search ~U`	alle aktualisierbaren Pakete anzeigen
	`aptitude search ?upgradable`	alle aktualisierbaren Pakete anzeigen
DNF	`dnf list upgrades`	alle aktualisierbaren Pakete anzeigen
`RPM`		
`YUM`	`yum list updates`	alle aktualisierbaren Pakete anzeigen

46.12 Verfügbare Pakete anzeigen

Welche Pakete verfügbar sind, erfahren Sie mit den nachfolgend genannten Aufrufen. In Abschnitt ``Bekannte Paketnamen auflisten`` Abschnitt 8.3 stellen wir Ihnen das genauer vor.

Tabelle 46.11: Verfügbare Pakete anzeigen

Werkzeug	Aufruf	Anmerkungen
APT	`apt-cache pkgnames`	alle verfügbaren (bekannten) Pakete auflisten
DNF	`dnf list available`	alle verfügbaren Pakete anzeigen
`RPM`		
`YUM`	`yum list available`	alle verfügbaren Pakete anzeigen

46.13 Paketsignatur überprüfen

Mit den nachfolgenden Aufrufen überprüfen Sie die Signatur eines Pakets. Sie stellen damit sicher, dass das Paket unverändert vom Paketmirror zu Ihnen übertragen wurde und auf dem Transportweg keine inhaltlichen Veränderungen stattgefunden haben. Für Debianpakete widmen wir uns dem Thema in „Paket verifizieren“ in Abschnitt 8.35 und „Paket auf Veränderungen prüfen“ in Abschnitt 8.29.

Tabelle 46.12: Paketsignatur überprüfen

Werkzeug	Aufruf	Anmerkungen
APT		
DNF		
`RPM`	`rpm -K` *Paketname*	
	`rpm --checksig` *Paketname*	
`YUM`		

46.14 Paket auf Veränderungen prüfen

Um festzustellen, ob die vorliegenden Dateien noch identisch mit den Dateien aus dem installierten Paket sind, helfen Ihnen diese Kommandos:

Tabelle 46.13: Paket auf Veränderungen prüfen

Werkzeug	Aufruf	Anmerkungen
`dpkg`	`dpkg -V`	prüft alle installierten Pakete
	`dpkg --verify` *Paketname*	prüft nur das angegebene Paket
`RPM`	`rpm -Va`	prüft alle installierten Pakete
	`rpm -qV` *Paketname*	prüft nur das angegebene Paket
	`rpm -Vp` *Paketname*	prüft nur das angegebene Paket
`YUM`		

APT und `aptitude` stellen keine eigenen Schalter zur Verfügung, `dpkg` erst ab der Version 1.17 (verfügbar ab Debian 8 *Jessie*). Für vorhergehende Veröffentlichungen weichen Sie auf die Werkzeuge `debsums` und `dlocate` aus. Darauf gehen wir im Abschnitt „Paket auf Veränderungen prüfen“ in Abschnitt 8.29 genauer ein.

Kapitel 47

Paketformat im Einsatz

47.1 Embedded-Geräte

Emdebian
Debian für Embedded Devices [Emdebian]

Raspbian
Debian speziell für den Raspberry Pi [Raspbian]

47.2 Bildung

Skolelinux
früher Debian-Edu, für den Einsatz in Schulen [Skolelinux]

Edubuntu
Ubuntu für Schule und Bildung [Edubuntu]

LernStick
Live-Distribution auf einem USB-Stick [LernStick]

47.3 Desktop

LiMux
Linux-Distribution, die im Rahmen der Migration der Stadtverwaltung München zum Einsatz kommt [LiMux]

Linux Mint, Linux Mint Debian Edition (LMDE)
basiert auf Ubuntu bzw. Debian *testing* [LinuxMint]

Tanglu
Rein community-basiertes Desktop-Linux [Tanglu]

Ubuntu (und dessen Ableger)
Linux für Einsteiger und den Desktop [Ubuntu]

Univention Corporate Server (UCS)
Linux mit integriertem Managementsystem für die zentrale und plattformübergreifende Verwaltung von Servern, Diensten, Clients, Desktops und Benutzern sowie von unter UCS betriebenen virtualisierten Computern [UCS]

47.4 Live-CD

Aptosid
Linux-Distributionen für Desktop-Computer und Notebooks, basierend auf Debian *unstable* [Aptosid]

BackTrack, Kali Linux
gedacht zur Netzwerk- und Sicherheitsanalyse. BackTrack [BackTrack] wurde im Frühjahr 2013 eingestellt, Kali Linux [Kali-Linux] ist dessen Nachfolger

Debian Live System
Projekt zur Erstellung und Pflege von Debian-basierten Live-Systemen und Debian Live Images [DebianLiveSystem]

Grml
textbasiert, gedacht als Rettungssystem und für die Systemanalyse, basiert auf Debian *unstable* [Grml]

Knoppix
basiert auf einer Mischung aus Debian *stable*, *unstable* und *testing* [Knoppix]

Siduction
Linux-Distributionen für Desktop-Computer und Notebooks, basierend auf Debian *unstable* [Siduction]

Zumindest Aptosid, Siduction, Grml und Knoppix lassen sich auch auf der Festplatte installieren und sind damit nicht nur Live-CDs.

47.5 Minimalsysteme

Damn Small Linux
wird seit Mitte 2012 nicht mehr weiterentwickelt [DamnSmallLinux]

47.6 Spieleplattform

Steam OS
ausgelegt für Spiele und die Verwendung großer Bildschirme [SteamOS]

47.7 Mobile Architekturen

Desweiteren kommt das `deb`-Paketformat auf mobilen Architekturen und Plattformen zum Einsatz. Dazu zählen etwa die Mobiltelefone Nokia N900 mit Maemo [Maemo], Nokia N9 mit Meego [MeeGo] sowie diverse Distributionen für das OpenMoko [OpenMoko]. Auch der community-getragene, inoffizielle N900-Nachfolger Neo900 [Neo900] soll u.a. mit Maemo und damit ebenfalls mit dem `deb`-Paketformat laufen.

47.8 Anstatt Linux

Auch mit Nicht-Linux-Kerneln wird das Paketformat eingesetzt. Einerseits gibt es Debian offiziell auch mit BSD- und GNU-Hurd-Kernel in Form von Debian GNU/kFreeBSD [Debian-Wiki-Debian-GNUkFreeBSD] und Debian GNU/Hurd [Debian-Wiki-Debian-GNUHurd]. Andererseits gibt es außerhalb des Debian-Projektes Portierungen auf den OpenSolaris bzw. Illumos-Kernel, z.B. Nexenta OS [NexentaOS], Illumian [Illumian] und Dyson OS [DysonOS]. Das mit Nexenta OS verwandte StormOS [StormOS] wurde allerdings bereits 2012 wieder eingestellt.

Unter Mac OS X existieren mit Fink [Finkproject] zusätzliche, freie Pakete. Diese können Sie über einen Jailbreak auch auf Ihrem iPhone, iPod und iPad benutzen.

47.9 Nachbauten und Derivate

Gerade in der Embedded-Szene, wo es auf Kompaktheit ankommt, sind `dpkg` und APT oft zu groß und komplex. Dennoch sind die Grundideen von Debians Paketmanagement-System auch in dieser Community beliebt und werden genutzt. So ist das *Itsy Package Management System* (`ipkg`) [ipkg] und später dessen Fork *OpenMoko Package Management System* (`opkg`) [opkg] entstanden. `opkg` ist heute noch u.a. bei OpenWrt im Einsatz, einer bekannten Linux-Distribution für WLAN-Router.

Auch Canonical – das Unternehmen hinter Ubuntu – versucht sich seit 2013 in einem Derivat von Debians Paketsystem. Ihre sogenannten *Click-Pakete* (siehe [Click-Paket-Format] und [SingleClickInstall]) funktionieren ähnlich wie `deb`-Pakete, jedoch ohne große Abhängigkeiten, und sind optimiert auf den Einsatz bei mobilen Geräten von Drittanbietern. Die hervorgehobenen Merkmale sind die direkte Installation des Pakets aus dem Webbrowser (siehe auch Kapitel 24) und die geringen Paketabhängigkeiten. Das Ziel besteht darin, alle benötigten Daten einer Software in möglichst einem Paket bereitzustellen.

Wie sich in der Diskussion zeigt, ist der Einsatz der Click-Pakete recht umstritten (siehe [Click-Paket-Format-Diskussionen] und [Watson-App-Design]). Mittlerweise ist dieses Format vor dessen größerer Verbreitung bereits durch den seit Herbst 2015 verwendeten Nachfolger Snappy [Ubuntu-Snappy] [Ubuntu-Snappy-Projekt] überholt.

47.10 Weitere Debian-Derivate

Einen ausführlichen Überblick zu weiteren Debian-Derivaten gibt der Debian-Derivate-Zensus. Er ist ein Bestandteil des Debian-Wikis [DebianDerivativeCensus].

Kapitel 48

Index

B

E