Debian-Paketmanagement

Axel Beckert und Frank Hofmann

Onyx Neon
<table>
<thead>
<tr>
<th>NUMMER</th>
<th>DATUM</th>
<th>BESCHREIBUNG</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>debian/0_2023.03.11-218-ge022c6b</td>
<td>2024-01-23T09:19:10+00:00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Inhaltsverzeichnis

1 Konzepte

1.1 Was ist Debian? 2
1.2 Debian-Architekturen
   1.2.1 Debian-Ports-Projekt 3
   1.2.2 Pakete für alle Architekturen 3
   1.2.3 Multiarch: Mehrere Architekturen gleichzeitig auf einem System 4
   1.2.4 Bevor es Multiarch gab 5
1.3 Vom \texttt{tar.gz} zur Linux-Distribution 5
1.4 Debians Paketsystem 5
1.5 Welche UNIX-artigen Betriebssysteme verwenden das Paketformat und das APT-Paketmanagement 6

2 Software in Paketen organisieren

2.1 Was ist Paketmanagement 7
2.2 Varianten und Formate für Softwarepakete 8
2.3 Softwarestapel und Ebenen 10
   2.3.1 Ebenen 10
   2.3.2 Untere Ebene 10
   2.3.3 Obere Ebene 11
   2.3.4 Paketformate und -werkzeuge anderer Distributionen 11
   2.3.5 Werkzeuge, die verschiedene Paketformate unterstützen 11
2.4 Alternativen zu APT 11
2.5 Zusammenspiel von dpkg und APT 12
2.6 Vom monolithischen Programm zu Programmkomponenten 14
2.7 Debi ans-Pakete (Varianten) 15
   2.7.1 Binärpakete (\texttt{deb}) 15
   2.7.2 Übergangspakete, Metapakete und Tasks 16
   2.7.3 Mikro-Binärpakete 17
   2.7.4 Source-Pakete (\texttt{dsc} und weitere Dateien) 17
Debian-Paketmanagement

2.16 Wie finde ich passende Pakete ........................................... 44
  2.16.1 Paketquellen .......................................................... 44
  2.16.2 Paketnamen ............................................................ 45
  2.16.3 Paketeigenschaften und Einordnung ............................... 45

II Werkzeuge ........................................... 46

3 Paketquellen und Werkzeuge ........................................... 47
  3.1 Paketquellen ............................................................ 47
    3.1.1 Begriff und Hintergrund ....................................... 47
    3.1.2 Benutzte Paketquellen ......................................... 47
    3.1.3 Aufbau und Struktur einer Paketquelle ....................... 47
  3.2 Empfehlung zum Ablauf für das Hinzufügen und Ändern von Paketquellen ........................................... 47
  3.3 Die Datei /etc/apt/sources.list verstehen .......................... 48
    3.3.1 Format der Paketliste .......................................... 48
    3.3.2 Format eines Eintrags .......................................... 49
    3.3.3 Beispieleinträge für offizielle Pakete ....................... 50
    3.3.4 Verzeichnis als Paketquelle ................................... 51
    3.3.5 Einträge für Sicherheitsaktualisierungen .................... 51
    3.3.6 Einträge für zusätzliche, nicht-offizielle Pakete ........... 52
    3.3.7 Einträge für Quellpakete ...................................... 52
    3.3.8 Einträge für Deutschland ...................................... 53
  3.4 Geeigneten Paketmirror auswählen ................................... 53
    3.4.1 Paketmirror bei Debian ........................................ 53
    3.4.2 Paketmirror für andere Distributionen ....................... 54
    3.4.3 Pakete ohne Paketmirror beziehen ............................. 55
  3.5 Am besten erreichbaren Paketmirror finden ........................ 55
    3.5.1 netselect und netselect-apt ................................... 55
      3.5.1.1 Paketquellen nach Pingzeiten und Entfernung auswählen ...... 56
      3.5.1.2 Anzahl der Hops begrenzen ................................ 58
      3.5.1.3 Einen geschützten Paketmirror abfragen .................. 59
      3.5.1.4 Liste der Paketquellen mit netselect-apt generieren lassen 59
      3.5.1.5 netselect und netselect-apt im Alltagseinsatz ............ 62
  3.6 Automatisiertes Auswählen von Paketquellen ........................ 62
    3.6.1 DNS Round Robin ................................................. 62
    3.6.2 Paketquellen über GeoIP auswählen ............................ 62
## 3.6.3 Per CDN

3.6.4 Automatische Paketmirror-Auswahl per Mirror-Liste

3.6.5 Welcher Paketmirror wird schlussendlich benutzt?

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

#### 3.8 Physische Installationsmedien mit apt-cdrom einbinden

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

##### 3.9.1 Aufruf und Optionen

##### 3.9.2 Beispiele

### 3.10 Einstellungen mit Synaptic

### 3.11 Debian und Ubuntu Sources List Generator

#### 3.11.1 Feinheiten für Debian

#### 3.11.2 Feinheiten für Ubuntu

### 3.12 Paketquelle auf Echtheit überprüfen

#### 3.12.1 Basiswissen

#### 3.12.2 Schlüsselverwaltung mit apt-key (Überblick)

#### 3.12.3 Unterkommandos von apt-key

#### 3.12.4 Beispiel: Ergänzung eines Schlüssels

#### 3.12.5 Abkündigung von apt-key

#### 3.12.6 Alternative Benutzerschnittstellen zur APT-Schlüsselverwaltung

### 3.13 Liste der verfügbaren Pakete aktualisieren

#### 3.13.1 Grundlegendes Vorgehen

#### 3.13.2 Überprüfung der Paketsignaturen

#### 3.13.3 Platz für den Paketcache

#### 3.13.4 Die Veröffentlichung wechseln

### 3.14 Lokale Paketliste und Paketcache

### 3.15 Lokale Paketliste reparieren

#### 3.15.1 Aktualität des Mirrors überprüfen

### 4 Debian-Paketformat im Detail

#### 4.1 Konzepte und Ideen dahinter

##### 4.1.1 Binärpakete

##### 4.1.2 Sourcepakete

##### 4.1.3 Weitere Metadaten

#### 4.2 Aufbau und Format

##### 4.2.1 Generell: 2 Ebenen

##### 4.2.2 Source-Pakete

##### 4.2.3 Binärpakete

#### 4.2.3.1 Komponenten

#### 4.2.3.2 Benennung

#### 4.2.3.3 Steuerdateien und Skripte

#### 4.2.3.4 Daten im Paket

#### 4.2.4 Übergangs- und Metapakete
## 5 APT und Bibliotheken  

<table>
<thead>
<tr>
<th>5.1 Bibliothek libapt-pkg</th>
<th>5.2 Bibliothek libapt-pkg-perl</th>
<th>5.3 Bibliothek python-apt</th>
<th>5.4 Paket libapt-pkg-doc</th>
<th>5.5 Bibliothek libapt-inst</th>
</tr>
</thead>
</table>

## 6 Werkzeuge zur Paketverwaltung (Überblick)  

| 6.1 Frontends für das Paketmanagement |
|--------------------------------------|----------------------------------|
| 6.1.1 Aufgaben, Sinn und Zweck des Frontends |
| 6.1.2 Anmerkungen zur Programmauswahl |
| 6.2 Für die Kommandozeile |
| 6.2.1 dpkg |
| 6.2.2 APT |
| 6.2.2.1 Überblick |
| 6.2.2.2 Komponenten und Funktionen |
| 6.2.2.3 apt-cache |
| 6.2.2.4 apt-get |
| 6.2.2.5 apt-key und apt-mark |
| 6.2.2.6 Weiterentwicklung von APT |
| 6.2.3 wajig |
| 6.2.4 sysget |
| 6.2.5 Cupt |
| 6.3 ncurses-basierte Programme |
| 6.3.1 taskel |
| 6.3.2 aptitude |
| 6.3.3 Nala |
| 6.4 GUI zur Paketverwaltung |
| 6.4.1 Synaptic |
| 6.4.2 Muon |
| 6.4.3 Smart Package Management (SmartPM) |
| 6.4.4 PackageKit |
| 6.4.5 GDebi |
| 6.5 Webbaasierte Programme |
| 6.5.1 Ubuntu Landscape |
| 6.5.2 Appnr |
| 6.5.3 Communtu |
| 6.5.4 Univention Corporate Server (UCS) |
7 Paketcache

7.1 Hintergrundwissen ................................................. 120
  7.1.1 Was passiert, wenn nicht alle Pakete heruntergeladen werden konnten? .... 120
7.2 Dateien im Paketcache ............................................. 121
7.3 Paketcache-Status .................................................. 121
7.4 Größe des Paketcaches ............................................. 123
  7.4.1 Wie viel Platz belegt der Paketcache? ......................... 123
  7.4.2 Größe des Paketcaches festlegen ............................... 123
7.5 Paketcache aufräumen ............................................. 124
  7.5.1 Weshalb aufräumen? ........................................... 124
  7.5.2 Paketverwaltung passend konfigurieren ....................... 124
  7.5.3 Kommandos zum Aufräumen .................................... 125
  7.5.4 Empfehlungen zum Zeitpunkt des Aufräumens .................. 126
  7.5.5 Automatisch und regelmäßig Aufräumen ...................... 127

8 Paketoperationen ..................................................... 128

8.1 Paketoperationen und deren Abfolge ............................. 128
8.2 Paketlisten und Muster ............................................ 128
8.3 Bekannte Paketnamen auflisten ................................ 129
8.4 Paketstatus erfragen .............................................. 132
  8.4.1 dpkg -s Paketname und dlocate -s Paketname .................. 132
  8.4.2 dpkg -I deb-Datei ........................................... 133
  8.4.3 apt-cache show Paketname .................................... 133
  8.4.4 apt-cache showpkg Paketname ................................ 134
  8.4.5 aptitude show Paketname ..................................... 136
  8.4.6 Anfragen mit apt-mark ....................................... 136
8.5 Liste der installierten Pakete anzeigen und deuten .............. 137
  8.5.1 dpkg -l Paketname (Langform --list) ........................ 137
  8.5.1.1 Ausgabe von dpkg interpretieren .......................... 137
  8.5.1.2 Vollständige Paketliste anzeigen ........................... 138
  8.5.1.3 Liste anhand von Filten einschränken ...................... 139
  8.5.1.4 Ausgabe minimieren ....................................... 140
  8.5.2 Suche mit aptitude .......................................... 141
  8.5.3 apt list --installed ....................................... 141
    8.5.3.1 Vollständige Paketliste anzeigen ....................... 141
    8.5.3.2 Liste anhand von Filten einschränken ................... 142
    8.5.3.3 Ausgabe minimieren .................................... 142
  8.5.4 Weitere Möglichkeiten ....................................... 143
8.6 Liste der installierten Kernelpakete anzeigen .................. 143
8.7 Liste der installierten, nicht-freien Pakete anzeigen .................................................. 144
  8.7.1 Recherche mit aptitude ............................................................ 144
  8.7.2 Recherche mit check-dfsg-status (vormals vrms) ........................................ 144
8.8 Neue Pakete anzeigen .............................................................. 146
8.9 Pakete nach Prioritäten finden ......................................................... 146
8.10 Automatisch installierte Pakete anzeigen ................................................. 147
  8.10.1 apt-mark benutzen .............................................................. 147
  8.10.2 aptitude benutzen .............................................................. 148
8.11 Obsolete Pakete anzeigen ............................................................ 148
  8.11.1 Recherche auf der Kommandozeile ............................................. 149
  8.11.2 Recherche in graphischen Programmen ......................................... 150
  8.11.3 Umgang mit diesen Paketen .................................................... 150
8.12 Aktualisierbare Pakete anzeigen ....................................................... 150
  8.12.1 apt-get verwenden ................................................................. 150
  8.12.2 apt benutzen ................................................................. 151
  8.12.3 aptitude verwenden ........................................................... 151
  8.12.4 Synaptic verwenden ............................................................ 151
8.13 Verfügbare Versionen eines Paketes anzeigen .............................................. 152
  8.13.1 aptitude verwenden ............................................................. 152
  8.13.2 Mit apt stöbern ........................................................................ 153
  8.13.3 apt-show-versions verwenden ................................................ 153
  8.13.4 apt-cache benutzen ............................................................. 154
  8.13.5 rmadison einsetzen ................................................................. 154
  8.13.6 grep-available und grep-aptavail benutzen .................................... 155
8.14 Aus welchem Repo kommen die Pakete ...................................................... 155
  8.14.1 Paketquellen untersuchen mit apt-cache policy ......................... 155
  8.14.2 Informationen für ein bestimmtes Paket erhalten ................................ 156
  8.14.3 Verfügbare Paketversionen ermitteln .................................... 157
8.15 Installationsgröße eines Pakets ........................................................................ 157
8.16 Größtes installiertes Paket finden .............................................................. 158
  8.16.1 dpkg ................................................................. 158
  8.16.2 dpigs ........................................................................ 158
  8.16.3 aptitude ................................................................. 159
8.17 Warum ist ein Paket installiert ........................................................................ 160
8.18 Liste der zuletzt installierten Pakete anzeigen ............................................ 162
  8.18.1 Statusdaten von dpkg ............................................................ 163
  8.18.2 Statusdaten von apt ............................................................. 164
8.19 Paketabhängigkeiten anzeigen ................................................................... 164
  8.19.1 Die Abhängigkeiten eines Pakets auflisten ........................................ 164
8.19.1.1 Ausgabe der Paketabhängigkeiten mit dpkg-deb ............................................. 165
8.19.1.2 Ausgabe der Paketabhängigkeiten mit apt-cache ............................................. 165
8.19.1.3 Recherche mit apt-rdepends ................................................................. 167
8.19.1.4 Ausgabe der Paketabhängigkeiten mit aptitude ............................................. 168
8.19.1.5 Ausgabe der Paketabhängigkeiten mit grep-status ............................................. 169
8.19.2 Anzeige der umgekehrten Paketabhängigkeiten .................................................. 169
8.19.2.1 Recherche mit apt-cache ............................................................................. 170
8.19.2.2 Recherche mit apt-rdepends ....................................................................... 170
8.19.2.3 Recherche mit aptitude ............................................................................... 171
8.19.3 Prüfen, ob die Abhängigkeiten des gesamten Systems erfüllt sind ..................... 171
8.19.4 Zusammenfassung aller unerfüllten Abhängigkeiten im Paketcache .................... 171
8.20 Pakete über den Namen finden ................................................................................. 172
8.20.1 Systemwerkzeuge ............................................................................................... 172
8.20.1.1 dpkg ........................................................................................................... 172
8.20.1.2 APT und apt-cache ...................................................................................... 173
8.20.1.3 aptitude ....................................................................................................... 173
8.20.1.4 grep-available und grep-status .................................................................. 174
8.20.1.5 Synaptic ....................................................................................................... 175
8.20.1.6 SmartPM ..................................................................................................... 176
8.20.2 Browserbasierte Suche ....................................................................................... 176
8.20.2.1 In Paketen blättern mittels dpkg-www ................................................................ 176
8.20.2.2 Suche über die Webseite des Debian-Projekts ............................................ 179
8.20.2.3 Suche über die Webseite von Debian-Derivaten ............................................ 181
8.20.2.4 Suche über apt-browse.org .......................................................................... 183
8.20.2.5 Suche über apt-get.org ............................................................................... 183
8.20.2.6 Rpmseek.com .............................................................................................. 185
8.21 Pakete über die Paketbeschreibung finden ............................................................... 185
8.21.1 Suche mit apt-cache ........................................................................................... 185
8.21.2 Suche mit Hilfe von aptitude ............................................................................. 187
8.21.3 Suche mit grep-available und grep-status ....................................................... 187
8.22 Paket nach Maintainer finden .................................................................................. 188
8.22.1 Welche Pakete betreut ein Debian-Maintainer ................................................... 188
8.22.2 Rückrichtung: Wer betreut ein bestimmtes Paket .............................................. 190
8.23 Paket nach Lizenz finden ........................................................................................ 191
8.24 Paket zu Datei finden .............................................................................................. 191
8.24.1 Suche in bereits installierten Paketen .................................................................. 191
8.24.2 Suche in noch nicht installierten Paketen ......................................................... 192
8.24.3 Suche über die Webseite des Debian-Projekts .................................................. 194
8.25 Paket auseinandernehmen ...................................................................................... 195
8.25.1 Mit ar in seine Bestandteile zerlegen ............................................. 195
8.25.2 Mit dpkg die Installationsstruktur herausfinden ............................... 196
8.26 Paketinhalte anzeigen ................................................................. 196
8.26.1 dpkg -L Paketname ....................................................................... 197
8.26.2 dlocate -L Paketname .................................................................... 197
8.26.3 dlocate -ls Paketname ................................................................. 197
8.26.4 dpkg -c deb-Datei ......................................................................... 197
8.26.5 apt-file show Paketname und apt-file list Paketname ...................... 198
8.26.6 Einsatz von dglob ........................................................................... 198
8.27 Nach Muster in einem Paket suchen ..................................................... 199
8.28 Ausführbare Dateien anzeigen ........................................................... 200
8.29 Manpages anzeigen ........................................................................... 200
8.29.1 Manpages erstöbern ....................................................................... 200
8.29.2 Manpages aus noch nicht installierten Paketen anzeigen .................. 201
8.29.3 Suche über den Webbrowser ........................................................... 203
8.30 Konfigurationsdateien eines Pakets anzeigen ....................................... 204
8.31 Paketänderungen nachlesen .................................................................. 206
8.31.1 Das Änderungsprotokoll beziehen .................................................... 206
8.31.2 Zwei Paketversionen miteinander vergleichen .................................. 208
8.32 Paket auf unerwünschte Veränderungen prüfen ..................................... 209
8.32.1 Prüfung eines Paketes auf Unversehrtheit .......................................... 209
8.32.1.1 Nur ein Einzelpaket kryptographisch prüfen ............................... 210
8.32.1.2 Mehrere Pakete prüfen .............................................................. 211
8.32.2 Die Inhalte eines bereits installierten Paketes überprüfen ................... 211
8.32.2.1 MD5-Summen zur Erkennung von Änderungen ....................... 212
8.32.2.2 MD5-Summen von Dateien mit dlocate anzeigen ..................... 212
8.32.2.3 Dateien paketbezogen mit dlocate überprüfen .......................... 212
8.32.2.4 Dateien überprüfen mit debsums .............................................. 213
8.32.2.5 Dateien mit dpkg -V überprüfen .............................................. 214
8.33 Liste der zuletzt geänderten Abhängigkeiten ....................................... 215
8.34 Paketdatei nur herunterladen ............................................................. 215
8.35 Installation zwischengespeicherter Pakete aus dem Paketcache .......... 217
8.36 Sourcepakete beziehen ...................................................................... 218
8.37 Sourcepakete anzeigen ...................................................................... 219
8.37.1 apt-cache verwenden ...................................................................... 219
8.37.2 apt-show-source verwenden .......................................................... 220
8.38 Pakete installieren .............................................................................. 220
8.38.1 Vorbereitungen ............................................................................... 220
8.38.2 Durchführung .................................................................................. 222
8.38.3 Begutachtung .............................................................. 222
8.38.4 Weitere, nützliche APT-Optionen für den Alltag (Auswahl) ................................. 223
8.38.5 Besonderheiten bei aptitude ....................................... 223
8.38.6 Erweiterungen ab APT 1.1 ........................................... 223
8.39 Pakete erneut installieren ............................................. 224
8.39.1 Wiederinstallieren vollständig entfernter Pakete ............................................. 224
8.39.2 Wiederinstallieren von Paketen mit vorhandenen Konfigurationsdateien ............... 224
8.39.3 Wiederinstallieren bereits installierter Pakete .............................................. 225
8.39.4 Typische Stolperfälle bei Wiederinstallieren mehrerer Pakete ......................... 225
8.40 Pakete konfigurieren ..................................................... 225
8.40.1 Bestehende Konfiguration eines Pakets anzeigen ........................................... 225
8.40.2 Konfiguration für alle Pakete auslesen ......................................................... 226
8.40.3 Bestehende Konfiguration anwenden ......................................................... 227
8.40.4 Konfiguration mit dpkg-reconfigure erneut durchführen .................................. 228
8.41 Pakete aktualisieren ..................................................... 229
8.41.1 update ................................................................. 230
8.41.2 install ............................................................... 230
8.41.3 upgrade und safe-upgrade ........................................ 231
8.41.4 dist-upgrade und full-upgrade ....................................... 232
8.41.5 Empfohlene Schrittfolge zur Aktualisierung von Paketen ................................. 232
8.41.6 Aktualisierung mit Synaptic ........................................ 233
8.42 Pakete downgraden ...................................................... 233
8.42.1 Hintergrund und Fragen zum Downgrade .................................................... 233
8.42.2 Ablauf und Durchführung ........................................ 234
8.42.2.1 Bestehende Paketversionen klären ......................................................... 234
8.42.2.2 Paket austauschen .................................................................................. 236
8.42.2.3 Paket über die Angabe der Versionsnummer austauschen ......................... 236
8.42.2.4 Paket über die Angabe der Veröffentlichung austauschen ......................... 237
8.43 Pakete deinstallieren ..................................................... 237
8.43.1 Fall 1: Paket einfach löschen ....................................... 237
8.43.2 Fall 2: Suche von Konfigurationsdateien bereits deinstallierter Pakete .......... 238
8.43.3 Fall 3: Löschen von Konfigurationsdateien bereits deinstallierter Pakete ........ 239
8.43.4 Fall 4: Paket samt Konfigurationsdateien deinstallieren ................................. 241
8.43.5 Fall 5: Paket für eine ausgewählte Architektur entfernen ................................. 241
8.44 Umgang mit Waisen ...................................................... 242
8.44.1 APT und aptitude ..................................................... 242
8.44.2 debfoster ............................................................ 243
8.44.3 deborphan ............................................................ 245
8.44.4 Orphaner und Editkeep .............................................. 247
8.44.5 gtkorphan ................................................................. 248
8.44.6 wajig ................................................................. 249
8.44.7 rpmorphan ............................................................. 249
8.45 Paketoperationen erzwingen ........................................ 250
  8.45.1 Aktionen mit dpkg erzwingen ................................ 250
  8.45.2 Aktionen mit apt erzwingen .................................... 252
  8.45.3 Aktionen an der Paketverwaltung vorbei ..................... 252
8.46 Paketstatusdatenbank reparieren .................................. 252
  8.46.1 Bit-Dreher reparieren ............................................ 252
  8.46.2 Die Paketstatusdatenbank aus dem lokalen Backup wiederherstellen ............................................................................................................ 253
  8.46.3 Die Paketstatusdatenbanken von APT und aptitude .......... 253
8.47 Distribution aktualisieren (update und upgrade) ............... 253
  8.47.1 Vorworte .............................................................. 253
  8.47.2 Vom upgrade zum dist-upgrade ................................ 254
  8.47.3 Unsere empfohlene Reihenfolge ................................ 254
  8.47.4 Anmerkungen ........................................................ 255

9 Dokumentation ............................................................ 256
  9.1 Man- und Infopages .................................................. 256
  9.2 Dokumentation in /usr/share/doc/ .............................. 256
  9.3 Die apt-dpkg-Referenzliste ....................................... 256
  9.4 apt-doc — das Benutzerhandbuch zu APT ...................... 257
  9.5 Der Debian-Spickzettel ............................................. 258
  9.6 APT-Spickzettel von Nixcraft ..................................... 259
  9.7 Pacman Rosetta ........................................................ 260
  9.8 Handbuch zu aptitude ................................................ 261
  9.9 The Debian Administrator’s Handbook ......................... 262
  9.10 Weitere Bücher ........................................................ 263

III Praxis ................................................................. 264

10 APT und aptitude auf die eigenen Bedürfnisse anpassen ....... 265
  10.1 Konfiguration von APT .............................................. 265
  10.2 Konfiguration von APT anzeigen .................................. 266
  10.3 Konfigurationsdateien von APT im Detail ...................... 268
    10.3.1 /etc/apt/apt.conf(.d) verstehen ......................... 268
    10.3.2 preferences bzw. preferences.d .......................... 269
    10.3.3 cron.daily/apt ................................................ 270
  10.4 Konfigurationsoptionen von APT ................................ 270
<table>
<thead>
<tr>
<th>Kapitel</th>
<th>Titel</th>
<th>Seitenzahl</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.5</td>
<td>APT-Hooks</td>
<td>270</td>
</tr>
<tr>
<td>10.6</td>
<td>Konfigurationsdateien von aptitude</td>
<td>270</td>
</tr>
<tr>
<td>10.7</td>
<td>aptitude: Interaktives Ändern von Optionen</td>
<td>271</td>
</tr>
<tr>
<td>10.8</td>
<td>aptitude Format Strings</td>
<td>271</td>
</tr>
<tr>
<td>10.9</td>
<td>Für aptitude die Ausgabebreite festlegen</td>
<td>273</td>
</tr>
<tr>
<td>10.10</td>
<td>Bei aptitude die Ausgabe sortieren</td>
<td>274</td>
</tr>
<tr>
<td>10.11</td>
<td>aptitude-Gruppierung</td>
<td>274</td>
</tr>
<tr>
<td></td>
<td>10.11.1 Kommandozeile</td>
<td>274</td>
</tr>
<tr>
<td></td>
<td>10.11.2 Textoberfläche</td>
<td>274</td>
</tr>
<tr>
<td></td>
<td>10.11.2.1 Anordnung und Breite der Spalten</td>
<td>275</td>
</tr>
<tr>
<td></td>
<td>10.11.2.2 Aufteilung auf dem Bildschirm</td>
<td>276</td>
</tr>
<tr>
<td>10.12</td>
<td>aptitude-Farbflache anpassen</td>
<td>276</td>
</tr>
<tr>
<td></td>
<td>10.12.1 Zwischen aptitude-Themes wechseln</td>
<td>276</td>
</tr>
<tr>
<td></td>
<td>10.12.2 Standardvorgaben</td>
<td>276</td>
</tr>
<tr>
<td></td>
<td>10.12.3 Eigene Farben vergeben</td>
<td>276</td>
</tr>
<tr>
<td>11</td>
<td>Mit aptitude Vormerkungen machen</td>
<td>279</td>
</tr>
</tbody>
</table>
|         | 11.1 Vormerkungen über die Kommandozeile durchführen | 279
|         | 11.2 Vormerkungen über die Textoberfläche durchführen | 280
|         | 11.3 Bestehende Vormerkungen anzeigen        | 280        |
|         | 11.4 Vormerkungen simulieren                 | 282        |
|         | 11.5 Vormerkungen wieder aufheben            | 283        |
|         | 11.6 Vormerkungen ausführen                  | 283        |
|         | 11.7 Risiken und Seiteneffekte               | 284        |
| 12      | APT und aptitude mischen                     | 285        |
|         | 12.1 Hintergrund                             | 285        |
|         | 12.2 Sollten Sie das überhaupt machen?      | 285        |
|         | 12.3 Was ist zu beachten, wenn Sie das machen | 286
|         | 12.4 Empfehlungen für Dokumentation und Beispiele | 286
| 13      | Erweiterte Paketklassifikation mit Debtags   | 287        |
|         | 13.1 Einführung                              | 287        |
|         | 13.2 Kurzinfo zum Debtags-Projekt           | 287        |
|         | 13.3 Webseite zum Projekt                    | 288        |
|         | 13.4 Debtags-Werkzeuge                       | 289        |
|         | 13.5 Vergebene Schlagworte anzeigen          | 291        |
|         | 13.5.1 Auf der Kommandozeile                | 291        |
|         | 13.5.2 Integration in aptitude               | 292        |
|         | 13.5.3 Graphische Programme                  | 292        |
13.5.4 Über den Webbrowser ......................................................... 292
13.6 Suche anhand der Schlagworte .............................................. 292
   13.6.1 Über die Kommandozeile ................................................ 292
      13.6.1.1 Suche mittels debtags .............................................. 292
      13.6.1.2 Suche mit axi-cache .............................................. 294
   13.6.2 Textoberfläche von aptitude ........................................... 295
   13.6.3 Graphische Programme ................................................ 295
   13.6.4 Suche über den Webbrowser .......................................... 296
13.7 Pakete um Schlagworte ergänzen .......................................... 298
13.8 Verwendetes Vokabular bearbeiten und erweitern ...................... 299
   13.8.1 Alle verfügbaren Schlagworte anzeigen ............................. 299
   13.8.2 Informationen zu Schlagworten anzeigen ............................ 299
   13.8.3 Schlagworte bearbeiten .............................................. 300
14 Mehrere Pakete in einem Schritt ändern .................................. 302
   14.1 Mit apt-get ................................................................. 302
   14.2 aptitude ................................................................. 302
15 Ausgewählte Pakete aktualisieren .......................................... 304
   15.1 Nur ein einzelnes Paket aktualisieren ................................ 304
      15.1.1 Auf der Kommandozeile .............................................. 304
      15.1.2 Über die Textoberfläche von aptitude ............................ 305
      15.1.3 Durchführung bei Synaptic ....................................... 305
   15.2 Aktualisierung mit Wechsel der Veröffentlichung ..................... 305
16 Ausgewählte Pakete nicht aktualisieren .................................. 306
   16.1 Auf der Kommandozeile .................................................. 306
   16.2 Textoberflächen ......................................................... 307
   16.3 Graphische Programme .................................................. 307
17 Fehlende Pakete bei Bedarf hinzufügen ................................... 308
   17.1 Neue Hardware ........................................................... 308
   17.2 Neue Software ........................................................... 309
      17.2.1 Empfehlungen mittels command-not-found ....................... 309
18 Alternative Standard-Programme mit Debians Alternativen-System .... 310
   18.1 Hintergrund: Warum alternative Standardprogramme? ............... 311
   18.2 Standardprogramme anzeigen ........................................... 312
   18.3 Standardprogramm ändern .............................................. 313
19 Backports 316

19.1 Ausgangssituation 316
19.2 Gegenüberstellung der verschiedenen Lösungsansätze 316
19.3 Debian Backports 317
19.4 Welche Pakete gibt es als offiziellen Backport? 317
19.5 Welche Versionen gibt es als offizielle Backports? 317
19.6 Einbindung in den Paketbestand 318
19.7 Die installierten Pakete anzeigen 318
19.8 Weiterführende Dokumentation 320
19.9 Backports bei Ubuntu 320
19.10 Wichtige Fragen, die sich bei Backports ergeben 320

20 Veröffentlichungen mischen 321

20.1 Die bevorzugte Veröffentlichung für alle Pakete festlegen 321
20.2 apt-get mit expliziter Angabe der Veröffentlichung 321
20.3 Von APT zu APT-Pinning 322
20.4 Paketweise festlegen 322
20.5 Praktische Beispiele 323

21 Pakete bauen mit checkinstall 324

21.1 Pakete aus zusätzlichen Quellen ergänzen 324
21.2 Software selbst übersetzen und einspielen 324
21.3 Software selbst übersetzen und als deb-Paket einspielen 324
21.4 Beispiel 326
21.5 Vor- und Nachteile 326
21.5.1 Weitere noch unbehandelte Notizen 327

22 Metapakete bauen 328

22.1 Vorbereitungen 328
22.2 Das Paket bauen 329
22.3 Die Komponenten des Pakets kryptographisch signieren 332
22.4 Das Debianpaket kryptographisch signieren 333
22.4.1 dpg-sig verwenden 333
22.4.2 debsigs benutzen 334
22.5 Das neue Paket benutzen 334
22.5.1 Mittels dpgkg und APT 334
22.5.2 Mittels gdebi 335
22.5.3 Mittels apt 336
23 Paketformate mischen

23.1 Einführung .................................................. 337
23.2 Fremdformathe mit alien hinzufügen
   23.2.1 Einführung ........................................... 337
   23.2.2 Pakete umwandeln ................................. 338
      23.2.2.1 Voraussetzungen .......................... 338
      23.2.2.2 Durchführung .................................. 338
      23.2.2.3 Fallstricke und Besonderheiten bei der Umwandlung ............................ 339
   23.2.3 Umgewandelte Pakete einspielen ............. 340
   23.2.4 Pakete umwandeln und einspielen ............. 341
   23.2.5 Fazit ................................................. 341
23.3 deb-Pakete in rpm-Strukturen ...................... 341

24 Umgang mit LTS ............................................ 342

24.1 Kurzzeitiges Abschalten der Gültigkeitsüberprüfung des Release Files ..................... 342
24.2 Dauerhaftes Abschalten der Gültigkeitsüberprüfung des Release Files ................. 342
24.3 Dienstleister zur Pflege veralteter LTS-Installationen .................................. 343

25 Webbasierte Installation von Paketen mit apturl ............................................. 344

25.1 Sinn und Zweck ............................................. 344
25.2 Risiken und Bedenken ................................. 345
25.3 apturl in der Praxis .................................... 345

26 Paketverwaltung beschleunigen ........................................ 346

26.1 Hintergrund ................................................. 346
26.2 Möglichkeiten zur Beschleunigung .......... 346
26.3 Empfehlungen zum Umgang im Alltag .......... 347

27 Paketverwaltung hinter einem http-Proxy .................................................. 348

27.1 Hintergrund .................................................. 348
27.2 Varianten ...................................................... 348
27.3 Einen Proxy konfigurieren ......................... 349
27.4 Einen Proxy mittels Umgebungsvariablen konfigurieren .......................... 349
27.5 Einen Proxy in Anwendungen konfigurieren .......... 349
   27.5.1 Mittels wget ........................................ 350
   27.5.2 Mittels curl .......................................... 350
   27.5.3 Webbrowser Firefox .............................. 351
   27.5.4 Webbrowser Chromium .......................... 352
27.6 APT über HTTP-Proxy ..................................... 352
   27.6.1 Konfigurationsdateien und Einstellungen .................. 352
   27.6.2 Schalter zur Steuerung des Cache-Verhaltens ................. 353
   27.6.3 Umgebungsvariablen ............................. 354
   27.6.4 Schalter für apt-get .............................. 354
27.7 Aptitude über HTTP-Proxy .................................. 354
28 Einen APT-Paket-Cache einrichten

28.1 Begriff

28.2 Besonderheiten des APT-Cache

28.2.1 Funktionsweise

28.2.2 Vor- und Nachteile

28.2.3 Abgrenzung zum Betreiben eines eigenen Paketmirrors

28.2.4 Softwareauswahl für einen APT-Cache

28.3 Approx

28.3.1 Überblick

28.3.2 Setup und Installation

28.3.3 Konfiguration

28.3.3.1 Server

28.3.3.2 Client

28.3.4 Beobachtungen aus dem Alltag

28.4 apt-cacher

28.4.1 Werkzeugbeschreibung

28.4.2 Einrichtung und Benutzung

28.4.2.1 Serverkonfiguration

28.4.2.2 Client anpassen

28.5 apt-cacher-ng

28.5.1 Werkzeugbeschreibung

28.5.2 Einrichtung und Betrieb

29 Cache-Verzeichnis auf separater Partition

29.1 Paketarchiv als tmpfs-Partition

29.2 Paketcache als separate Partition einrichten

29.3 Cache-Verzeichnis als Unterverzeichnis auf anderer Partition

30 Eigenes APT-Repository anlegen

30.1 Verzeichnis mit Paketen

30.2 dpkg-scanpackages

30.3 reprepro

30.4 mini-dinstall

30.4.1 Voraussetzungen

30.4.2 Einrichtung

30.4.2.1 Verzeichnisstruktur des APT-Repositorys

30.4.2.2 Konfiguration für mini-dinstall

30.4.2.3 Konfiguration für dput

30.4.2.4 Paket in das Repository hochladen
30.4.3 Laufender Betrieb ................................................................. 370
30.4.4 Schalter und Konfiguration ................................................ 371
30.4.5 Lesematerial ................................................................. 372
30.5 apt-ftparchive ................................................................. 372
30.6 aptly ................................................................. 373
30.7 debify ................................................................. 373
30.5 Einen eigenen APT-Mirror aufsetzen 374
31.1 apt-mirror ................................................................. 374
  31.1.1 Wichtige Dateien aus dem Paket ................................................. 375
  31.1.2 Ablauf ................................................................. 375
  31.1.3 Beispiel/HowTo ................................................................. 376
  31.1.4 Hinweise ................................................................. 376
31.2 debmirror ................................................................. 376
31.3 debpartial-mirror ............................................................. 377
31.4 ubumirror ................................................................. 377
31.5 debarchiver ................................................................. 377
31.6 reprepro ................................................................. 378
32 Plattenplatz sparen mit der Paketverwaltung 379
33 Platte läuft voll 380
  33.1 Hintergrund ................................................................. 380
  33.2 wie löst man diesen Zustand (Empfehlung zum Vorgehen) ................. 381
  33.3 Varianten ................................................................. 382
  33.4 Fehler beheben ................................................................. 382
34 Paketkonfiguration sichern 383
  34.1 Die bestehende Paketkonfiguration auslesen .................................. 383
    34.1.1 Mit dpkg ................................................................. 383
    34.1.2 Mit apt ................................................................. 384
    34.1.3 Mit debconf-get-selections ................................................. 384
    34.1.4 Mit apt-clone ................................................................. 385
  34.2 Eine gesicherte Paketkonfiguration wieder einspielen .................... 387
    34.2.1 Mit apt-get ................................................................. 387
    34.2.2 Mit debconf-set-selections ................................................. 387
    34.2.3 Mit apt-clone ................................................................. 388
  34.3 Graphische Werkzeuge ............................................................. 388
    34.3.1 Aptik ................................................................. 388
    34.3.2 Mintbackup ................................................................. 388
35 Automatisierte Installation

35.1 Einstieg ............................................. 390
35.2 Kriterien für die Auswahl einer geeigneten Lösung ........................... 391
35.3 Mit Debian-Werkzeugen .................................. 391
35.4 Komplette Lösungen .................................. 391
   35.4.1 Cobbler ........................................ 391
   35.4.2 FAI ............................................ 391
   35.4.3 Kickstart ...................................... 392
35.5 Erfahrungen aus der Praxis .................................. 392

36 Automatisierte Aktualisierung ........................................ 393

36.1 apt-dater ............................................. 393

37 Qualitätskontrolle ............................................. 394

37.1 Nicht installierte Pakete mit lintian prüfen ............................ 394
   37.1.1 lintian verstehen ................................ 394
   37.1.2 lintian verwenden ................................ 395
37.2 Bereits installierte Pakete mit adequate prüfen ............................ 397
37.3 Bugreports anzeigen ..................................... 399
   37.3.1 Hintergrundwissen ................................ 399
   37.3.2 Bugreports mit apt-listbugs lesen ............... 399
   37.3.3 Ergänzende Bugreports mit apt-listchanges herausfiltern ..... 400
   37.3.4 Release-kritische Fehler mit popbugs finden ........ 401
   37.3.5 Release-kritische Fehler mit rc-alert finden ...... 402
   37.3.6 Welche der von mir genutzten Pakete benötigen Hilfe? ........ 404
37.4 Auslaufende Sicherheitsaktualisierungen mit check-support-status anzeigen .... 405

38 Versionierte Paketverwaltung ......................................... 408

39 Pakete und Patche datumsbezogen auswählen .............................. 409

40 Paketverwaltung mit eingeschränkten Ressourcen für Embedded und Mobile Devices 410

40.1 Batterielaufzeit ....................................... 410
40.2 Speicherplatz ......................................... 410
40.3 localepurge .......................................... 410
40.4 Bildschirmgröße ....................................... 411
40.5 Erfordernisse an die CPU ................................. 411
41 Paketverwaltung ohne Internet 412

41.1 Hintergrund und Einsatzfelder 412

41.2 Strategien 412

41.2.1 Benötigte Pakete vorher explizit herunterladen 413

41.2.2 Einbindung fester Installationsmedien 413

41.2.3 Einbindung eines lokalen Paketmirrors 413

41.3 Werkzeuge 414

41.3.1 Offline-Verwaltung mit apt-get und wget 414

41.3.2 Das Projekt apt-offline 415

41.3.2.1 Projekttidea 415

41.3.2.2 Benutzung im Detail 415

41.3.3 Pakete mit dpkg-split aufteilen 416

41.3.4 Keryx 419

42 Systeme mit schlechter Internet-Anbindung warten 420

42.1 debdelta 420

42.2 PDiffs 423

43 Paketverwaltung hinter einer Firewall 424

44 Der APT- und aptitude-Wunschzettel 425

IV Ausblick 426

45 Notizen 427

46 Pakete selber bauen 428

47 Ein eigenes Debian-Repository aufbauen 429

48 Zukunft von APT, dpkg und Freunden 430

49 Fazit / Zusammenfassung 431

49.1 Was können Sie jetzt? 431

49.2 Empfehlungen für Einsteiger 431

49.2.1 Mit welchem Programm zur Paketverwaltung gelingt der Einstieg am leichtesten? 431

49.2.2 Installations-CD/DVD oder Netzwerkininstallation? 432

49.3 Empfehlungen für Fortgeschrittene und Profis 432
V Anhang

A Debian-Architekturen

A.1 Offizielle Architekturen
A.2 Architekturen, die nicht den Linux-Kernel verwenden
A.3 Veraltete Architekturen
A.4 Architekturen, deren Unterstützung vorgesehen ist

B Kommandos zur Paketverwaltung im Vergleich

B.1 Zusammenfassung
B.2 Die lokale Paketliste aktualisieren
B.3 Neue Softwarepakete installieren
B.4 Bereits bestehende Softwarepakete erneuern
B.5 Eine frühere Version eines Softwarepakets einspielen (Downgrade)
B.6 Ein Softwarepaket erneut installieren
B.7 Ein Softwarepaket nur herunterladen
B.8 Paket löschen / entfernen
B.9 Alle installierten Pakete auflisten
B.10 Einzelpaket auflisten
B.11 Abhängigkeiten anzeigen
B.12 Alle Dateien eines Pakets anzeigen, bevor es installiert wird
B.13 Alle Dateien eines installierten Pakets anzeigen
B.14 Alle Konfigurationsdateien eines Pakets anzeigen
B.15 Alle Dokumentationsdateien eines Pakets anzeigen
B.16 Paket identifizieren, aus dem eine Datei stammt
B.17 Paketstatus anzeigen
B.18 Aktualisierbare Pakete anzeigen
B.19 Verfügbare Pakete anzeigen
B.20 Paketsignatur überprüfen
B.21 Paket auf Veränderungen prüfen
B.22 Transaktionshistorie anzeigen
B.23 Eine installierte Veröffentlichung aktualisieren (Distributionsupgrade) ==

C Paketformat im Einsatz

C.1 Embedded-Geräte
C.2 Bildung
C.3 Desktop
C.4 Live-CD
C.5 Minimalsysteme
C.6 Spieleplattform
C.7 Mobile Architekturen
C.8 Anstatt Linux
C.9 Nachbauten und Derivate
C.10 Weitere Debian-Derivate
D Früher im Buch erwähnte Werkzeuge 454
E Literaturverzeichnis und Referenzen 456

50 Index 473
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.

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 von Entwicklern 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, Linux Mint, Knoppix oder 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


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.


Bevor wir uns daran machten, Passagen aus diesen umfangreichen Beiträgen wieder herauszuschneiden, 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 kaum welche. Wie in jedem größeren IT-Projekt gibt es neben den intensiv genutzten, gut dokumentierten Bereichen aber auch „dunkle Ecken“

\[Jörg, bitte nicht böse sein!\]
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. Hier im Buch zeigen wir Ihnen die verschiedenen (Schleich-)Wege, die wir kennen.


Das vorliegende Buch will darum vor allem Klarheit schaffen und Ihnen die Zusammenhänge zwischen den einzelnen Programmen deutlich machen. Es hilft Ihnen, in jeder Situation das passende Werkzeug zur Paketverwaltung auszuwählen und es danach gekonnt einzusetzen. Die einzelnen Kapitel sind aufgabenbezogen zusammengestellt. In jedem Abschnitt finden Sie Lösungen, wie Sie die jeweilige Aufgabe mit den verschiedenen Werkzeugen umsetzen.

Der Praxisteil fokussiert auf komplexere Fragestellungen. Dazu fassen wir den aktuellen Stand der Entwicklung zusammen und beleuchten darüber hinaus die angrenzenden Programme bzw. die damit verbundenen Situationen im Alltag der Systembetreuung.

**Baustellenstatus**


**Technische Basis**


---

**Versionsverwaltung mit Git**


---

**Online-Fassung**


Sollte die Ihnen vorliegende Fassung (sei es als Paket in einer Debian-Veröffentlichung oder als gedrucktes Buch) nicht aktuell genug sein, so schauen Sie doch mal in die Online-Fassung. Vielleicht wurde die entsprechende Stelle dort bereits aktualisiert.
Quellcode und Lizenz

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


Organisatorisch


Viele Texte verfassen wir zudem von unterwegs aus. Die bisherigen Stationen umfassen Aix-les-Bains, Ajacchio (Korsika), Ålesund (Norwegen), Andorra, Augsburg, Beauvais (Picardie), Bergneustadt, Berlin, Bern, Besançon, Biel/Bienne, Bottighofen (Bodensee, Schweiz), Bratland (bei Bergen, Norwegen), Bruchsali, Canterbury (Kent), Chemnitz, Cudrefin, Delémont, Edinburgh (Schottland), Engelberg-Titlis, Essen, Frankfurt/Main, Freiburg im Breisgau, Friedrichshafen, Genf, Gernersheim, Goizuetza (Baskenland, Spanien), Groß-Sankt-Bernhard-Paß, Hamburg, Hannover, Heidelberg, Hout Bay und Kapstadt (beide Western Cape, Südafrika), Kirchzarten bei Freiburg im Breisgau, Koblenz, Lauchringen (Baden, Wutachtal), Laveno Mombello (Lago Maggiore, Italien), Lausanne, London, Magdeburg, Meersburg (Bodensee), Montpellier, Montreux, München, Oldenburg in Oldenburg, Orø (Dänemark), Port del Cantó (Katalanische Pyrenäen, Spanien), Radebeul bei Dresden, Rømø (Dänemark), Rostock-Warnemünde, Saint-Cergue (Jura, Schweiz), Saint-Claude (Jura, Frankreich) Saint-Étienne, Saint-Jouin-Bruneval (Normandie), Saint-Victor-sur-Loire (Auvergne-Rhône-Alpes), Sankt Augustin (bei Bonn), Savines-le-Lac (Hautes Alpes, Frankreich), Insel Sokn (bei Stavanger, Norwegen), Tübingen, Tvinnefossen (Norwegen), Zernez (Engadin, Schweiz) und Zürich (siehe Abbildung 1). Orange Kreise mit rotem Rand markieren Axels Stationen, rote Kreise mit orangenem Rand die Arbeitsorte von Frank. Manchmal überlappt sich das auch — dann ist es nur ein von beiden. 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. Betreuen Sie Ihre Systeme als Benutzer in Eigenverantwortung, sind diese Kenntnisse für Sie im Alltag ebenso unverzichtbar.


Material für Ihre LPIC-Prüfungen

Ihre Vorbereitung auf die anspruchsvollen Tests des LPI ergänzen Sie am besten mit dem Buch „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, überholt 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], mittlerweile sogar auf der offiziellen Webseite von Debian.

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.


Dokumentation deb vs. rpm


Ein vergleichbares Buch zur Debian-basierten Paketverwaltung fehlte bislang. Viele hervorragende Kompendien (siehe dazu Abschnitt 9.10) 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 . . .


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, nennen wir diese und erklären, wie Sie in einem solchen Fall am besten verfahren.


Was Sie allerdings im vorliegenden Buch finden, ist die Zusammenstellung eines deb-Pakets — sprich: aus welchen Einzelteilen es besteht (siehe Abschnitt 4.2), wie Sie dieses in die Komponenten zerlegen (siehe Abschnitt 8.25) und auch wieder zusammenbauen (siehe Kapitel 21).

Zielgruppe und Lernziele


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


Die 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 geben darum aus, dass Sie auf einem Serversystem arbeiten und dieses ggfl. 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.


Sie müssen auf Ihrem System über administrative Benutzerrechte verfügen, um einen Großteil der Beispiele nachvollziehen zu können. Wir weisen nicht jedes Mal explizit darauf hin. In den Beispielen für die Kommandozeile erkennen Sie anhand des verwendeten Prompt-Zeichens, ob dafür administrative Rechte notwendig sind oder nicht: # bedeutet hierbei ja und $ bedeutet nein. 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].


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, d.h. installieren, aktualisieren und löschen
- 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
- alternative Auflösungen für Paketabhängigkeiten finden, verstehen und anwenden

All dies qualifiziert Sie für das entsprechende Lernziel einer Linux-Zertifizierung. 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 auch als Debian-Paket-Maintainer agieren zu können [Debian-Wiki-Debian-Entwickler].

Buchinfo

Wir pflegen eine buchbegleitende Webseite unter der URL:

https://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!

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

Etliche 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 gebührt 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`, `dpkg -L` und `dpkg -x`, die graphische Umsetzung der Landkarte, die vielen Anregungen zum Aufsetzen und Betreiben eines Proxies (siehe Kapitel 27) sowie dem Thema „Kryptographische Signaturen in Debian-Repositories“
• Mechtilde Stehmann (für die Sprachkorrekturen 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, Korrekturlesen rund um das Parsen von Debian-Versionsnummern und APT-Pinning sowie Konsistenzprüfung)
• Christoph Berg (für Tipps und Tricks rund um `reprepro` und seine Erfahrungen mit `apt.postgresql.org` [APT-Repository 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 24)
• Jörg Brühe für die Anregungen zu den Paketabhängigkeiten
• Sebastian Andres für seine Anregungen zu Debian Backports
• Gregor Herrmann für den Hinweis, das Buch auf Links zu alioth.debian.org-Webseiten hin zu überprüfen
• Alf Gaida für Hinweise auf nicht shell-unabhängige Beispiele
• Ingo Wichmann für die Hinweise zu den `rpm`- und `yum`-Kommandos

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.


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.


Zudem leiten sich aus Debian sehr viele Derivate für ausgewählte Zielgruppen oder Einsatzzwecke ab, z.B. Ubuntu, Linux Mint, Knoppix, Grml oder Dann 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


Architekturen, die das Werkzeug dpkg unterstützt (Ausschnitt)

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

Die Übersicht der Architekturen im Anhang (siehe Kapitel A) 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


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.


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


- 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/i686* mittels Hardware-Emulation und *arm*/jede Plattform (via Qemu Userland-Emulation).


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.


1.3 Vom tar.gz zur Linux-Distribution


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.


1.4 Debians Paketsytem


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 taskssel (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 6.4.4) – als Basis für Gnome-PackageKit und Apper bei KDE – sowie Muon (siehe Abschnitt 6.4.2), PackageSearch (siehe Abschnitt 13.4) 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


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.


Im Detail erklären wir Ihnen die Themmatik unter Pakete aktualisieren (siehe Abschnitt 8.41), Distribution aktualisieren (siehe Abschnitt 8.47), Paket downgraden (siehe Abschnitt 8.42) und dem Debian Backports Archiv (siehe Kapitel 19).


Ausgabe aller derzeit installierten Pakete für Abiword mit dpkg

$ dpkg -l "abiword*"
Gewünscht=Unbekannt/Installieren/R=Entfernen/P=Vollständig Löschen/Halten
<table>
<thead>
<tr>
<th>Status=Nicht/Installiert/Config/U=Entpackt/halb konFiguriert/</th>
<th>Fehler?=(kein)/R=Neuinstallation notwendig (Status, Fehler: GROSS=schlecht)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Architektur</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>abiword</td>
<td>2.9.2+svn20120 i386</td>
<td>efficient, featureful word processor</td>
<td></td>
</tr>
<tr>
<td>abiword-common</td>
<td>2.9.2+svn20120 all</td>
<td>efficient, featureful word processor</td>
<td></td>
</tr>
<tr>
<td>abiword-plugin-gram</td>
<td>2.9.2+svn20120 i386</td>
<td>grammar checking plugin for AbiWord</td>
<td></td>
</tr>
<tr>
<td>abiword-plugin-math</td>
<td>2.9.2+svn20120 i386</td>
<td>equation editor plugin for AbiWord</td>
<td></td>
</tr>
</tbody>
</table>


### 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_i18n_Translation-de
  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

Tabelle 2.1: Übersicht zu Paketformaten und deren Verbreitung

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

Seit 2007 bestehen Containerformate, die insbesondere mit VirtualBox und Docker populär wurden. Ziel ist, in diesen Containern bereits fertig installierte Anwendungen bereitzustellen. Dazu zählen bspw. die Formate Flatpak, OpenContainers, Linux Containers (LXC), Snappy und VirtualBox (VDI) (siehe [Docker], [Flatpak], [OpenContainer], [LXC], [Ubuntu-Snappy-Projekt] und [VirtualBox]).

Tabelle 2.2: Übersicht zu Containerformaten und deren Verbreitung

<table>
<thead>
<tr>
<th>Abkürzung</th>
<th>in Verwendung</th>
<th>Distribution</th>
<th>Name des Debianpaket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Docker</td>
<td>seit 2014</td>
<td>Debian, Ubuntu, RedHat/Fedora, openSUSE, CentOS</td>
<td>docker [Debian-Paket-docker]</td>
</tr>
<tr>
<td>Flatpak</td>
<td>seit 2015</td>
<td>RedHat/Fedora, Ubuntu, CentOS</td>
<td>flatpak [Debian-Paket-flatpak]</td>
</tr>
<tr>
<td>Linux Containers (LXC)</td>
<td>seit 2008</td>
<td>Debian, Ubuntu, RedHat/Fedora, CentOS</td>
<td>lxc [Debian-Paket-lxc]</td>
</tr>
<tr>
<td>OpenContainers</td>
<td>seit 2015</td>
<td>Debian, Ubuntu, RedHat/Fedora, CentOS</td>
<td>oci-image-tool [Debian-Paket-oci-image-tool]</td>
</tr>
<tr>
<td>Snappy</td>
<td>seit 2015</td>
<td>Ubuntu</td>
<td>nicht vorhanden</td>
</tr>
<tr>
<td>VirtualBox (VDI)</td>
<td>seit 2007</td>
<td>Debian, Ubuntu, RedHat/Fedora, openSUSE, CentOS, Oracle Linux</td>
<td>virtualbox (kein offizielles Debianpaket)</td>
</tr>
</tbody>
</table>

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


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 8.26) sowie `-l` zur Auflistung der installierten Pakete (siehe Abschnitt 8.5), `-s` zum Erfragen des Paketstatus (siehe Abschnitt 8.4) und `-S`, um das Paket zu finden, in dem eine bestimmte Datei vorkommt (siehe Abschnitt 8.24).

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 9.7) sowie in unserer Übersicht im Anhang des Buches (siehe Kapitel B).
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), Muon (siehe Abschnitt 6.4.2) oder auch PackageKit (siehe Abschnitt 6.4.4) installiert. Die Auswahl variiert und hängt von der von Ihnen gewählten Linux-Distribution und ihren Vorlieben ab.


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


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 (SmartPM) (siehe Abschnitt 6.4.3) und PackageKit (siehe Abschnitt 6.4.4). 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 6.2.3). 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

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.


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. Deshalb tauchen sie 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 hinzu. Abbildung 2.2, Abbildung 2.3 und Abbildung 2.4 zeigen die minimalen Paketabhängigkeiten für APT, dpkg und aptitude in graphischer Form.

Erzeugung der Abhängigkeitsgraphen zu dpkg mittels debtree

```bash
$ 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, label="(>= 5.1.1alpha+20120614)"];
  "dpkg" -> "liblzma5" [color=purple,style=bold, label="(>= 5.6+20070908)", label="(>= 5.6+20070908)"];
  "libboost-iostreams1.49.0" -> "libpcre3" [color=blue, label="(>= 8.10)"];
  "dpkg" -> "tar" [color=purple,style=bold, label="(>= 1.23)"];
```


Erzeugung der Abhängigkeitsgraphen und Speicherung als Rastergrafik

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

## 2.6 Vom monolithischen Programm zu Programmkomponenten


### Tabelle 2.3: Paketierung der Komponenten am Beispiel von APT

<table>
<thead>
<tr>
<th>Paketname</th>
<th>Paketkategorie</th>
<th>Komponente und Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt</td>
<td>Administration (admin)</td>
<td>Paketmanager für die Kommandozeile (siehe Abschnitt 6.2.2)</td>
</tr>
<tr>
<td>apt-doc</td>
<td>Dokumentation (doc)</td>
<td>Dokumentation zum Paket <code>apt</code></td>
</tr>
</tbody>
</table>
Tabelle 2.3: (continued)

<table>
<thead>
<tr>
<th>Paketname</th>
<th>Paketkategorie</th>
<th>Komponente und Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt-transport-https</td>
<td>Administration (admin)</td>
<td>APT-Plugin für HTTPS-Support (obsolet seit APT 1.5)</td>
</tr>
<tr>
<td>apt-utils</td>
<td>Administration (admin)</td>
<td>Hilfsprogramme für APT</td>
</tr>
<tr>
<td>libapt-instX.Y</td>
<td>Bibliotheken (libs)</td>
<td>Laufzeitbibliothek zum Paketformat</td>
</tr>
<tr>
<td>libapt-pkg.X.Y</td>
<td>Bibliotheken (libs)</td>
<td>Laufzeitbibliothek zur Paketverwaltung</td>
</tr>
<tr>
<td>libapt-pkg-dev</td>
<td>Bibliotheken zur Entwicklung (libdevel)</td>
<td>Entwicklerdateien zu libapt-pkg</td>
</tr>
<tr>
<td>libapt-pkg-doc</td>
<td>Dokumentation (doc)</td>
<td>Dokumentation zur Laufzeitbibliothek libapt-pkg</td>
</tr>
<tr>
<td>libapt-pkg-perl</td>
<td>Bibliotheken (libs)</td>
<td>Laufzeitbibliothek zur Paketverwaltung, Perl-Schnittstelle</td>
</tr>
</tbody>
</table>

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.


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).


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. Statt dessen liefern diese Abhängigkeiten auf andere Pakete.


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 6.5.3 ein.


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.

Metapakete selber bauen

Wie Sie Ihre eigenen Metapakete erstellen und diese dann auch zum Bezug in einem Repository bereithalten, lernen Sie unter „Metapakete bauen“ in Kapitel 22.
2.7.3 Mikro-Binärpakete

Mikro-Binärpakete tragen die Dateiendung udeb, wobei das u den griechischen Buchstaben Mu („µ“) repräsentiert. Sie sind technisch keine gewöhnlichen Binärpakete, sondern aufs Kleinstes heruntergestutzte Pakete. Sie kennen nur eine einzige Art von Paketrelation namens „hängt ab von“. Desweiteren beinhalten sie keine Maintainer-Skripte und führen auch sonst kaum Metainformationen mit.


2.7.4 Source-Pakete (dsc und weitere Dateien)

Diese Pakete beinhalten den Quellcode von Programmen und tragen das Suffix dsc als Abkürzung für Debian Source Control. Die Bestandteile eines solchen Paketes sind:


- eine Datei mit den Metadaten (Größe, Hashsummen, etc.) über die vorher genannten Dateien. Genutzt wird die Dateiendung 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


2.7.5 Virtuelle Pakete


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.


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 fettgedrucktem Text und der Markierung i für „Paket ist installiert“ in der ersten Spalte der Darstellung (siehe auch Abschnitt 6.2.1 für weitere Darstellungsvarianten).
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.


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 37.3 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


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 Anzeige hierarchie konfigurieren.

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 (otherofs)
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 (arduin, 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
[GOobject-Introspection] (gir1.2-ebook-1.2)

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

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

KDE (kde)
Programme zum KDE-Desktop (apper, kdm, knots)

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, ocamlwsc*)

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 (*freeeiv-server, gcompris, openentd*)

Sprachpakete *(localization)*
Lokalisierungsunterstützung für große Softwarepakete (*firefox-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, xdvi, 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, firefox*)
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**)
nachrichten rund um das Zope-Framework (**zope-common**, **zope2.13**)

---

**Erweiterung um Debtags**

---

**2.9 Distributionsbereiche**


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.

**non-free-firmware**
Firmware, die nicht den Debian-Richtlinien für freie Software (DFSG) entspricht, aber frei verteilbar ist. (Diesen Bereich gibt es erst ab Debian 12 Bookworm. Er wurde nach einer Abstimmung innerhalb von Debian im Jahre 2022 von non-free abgetrennt [gr-non-free-firmware]. In vorherigen Veröffentlichungen waren unfreie Firmware-Pakete im Bereich non-free untergebracht.)

---

**2.9.1 Einordnung der Distributionsbereiche in Debian**

Obwohl vielfach von außen anders wahrgenommen, zählt zur Debian-Distribution nur der Bereich **main**. Die anderen drei 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 drei anderen Bereiche **contrib**, **non-free** und/oder **non-free-firmware** dazuzunehmen.

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.

In den beiden Bereichen non-free und non-free-firmware 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 sind:

- Der Quellcode liegt nicht (komplett) vor.
- 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 diesen Bereichen ist es ratsam, immer erst anhand der Lizenz zu überprüfen, ob Sie diese Software überhaupt für Ihre gewünschten Zwecke einsetzen dürfen.

Für Software aus den beiden Bereichen non-free und non-free-firmware 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 aus einer anderen Quelle und danach manuell auf dem System eingepflegt.


2.9.2 Einordnung der Distributionsbereiche bei anderen Distributionen

Im Vergleich zu Debian sind bei Ubuntu die Distributionsbereiche etwas anders eingeteilt. 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). Zusätzlich existiert der Distributionsbereich *partner*, welcher für die Bereitstellung kommerzieller Software gedacht ist, deren Quellcode nicht offen liegt.


### 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.


### 2.9.4 Softwareverteilung

Bezogen auf die Anzahl der verfügbaren Softwarepakete findet sich der überwiegende Teil der Pakete im Bereich *main*, danach folgen *contrib*, *non-free* und *non-free-firmware*.

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.


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. oldstable, stable, testing oder unstable
- nach ihrem Alias-Namen (siehe Abschnitt 2.10.2), z.B. Wheezy, Jessie oder Stretch

Welche Veröffentlichung Sie auf Ihrem System verwenden, entnehmen Sie der Datei /etc/debian_version wie folgt:

**Die genutzte Debian-Version anzeigen**

```
$ cat /etc/debian_version
9.6
```

Ausführlichere Informationen erhalten Sie mit Hilfe des Kommandos lsb_release -a (Langform --all) aus dem Debian-paket lsb-release [Debian-Paket-lsb-release]:

**Ausführliche Informationen zur genutzten Debian-Version mit Hilfe von lsb_release anzeigen**

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description: Debian GNU/Linux 9.6 (stretch)
Release: 9.6
Codename: stretch
```


Abbildung 2.8: neofetch im Einsatz auf Debian 11 Bullseye

2.10.1 Bedeutung der verschiedenen Entwicklungsstände

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

**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**

**oldstable**

**oldoldstable**

**experimental**


### 2.10.2 Alias-Namen


- Debian 1.0 wurde nie offiziell veröffentlicht, da ein CD-Verteiler bedauerlicherweise eine Entwicklungsversion als Version 1.0 bezeichnet hatte [Debian-Project-History]. Daher entschlossen sich Debian und der CD-Verteiler zur gemeinsamen Bekanntmachung, dass die beigefügte Version fehlerhaft war ("this release was screwed") und das Projekt veröffentlichte die Version 1.1 ein halbes Jahr später.
- 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 3.1 *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* (14. Februar 2009; benannt nach dem aufziehbaren Ferngläs)
• Debian 6.0 *Squeeze* (6. Februar 2011; benannt nach den grünen dreigliedrigen Aliens)
• Debian 8 *Jessie* (25. April 2015; benannt nach der jodelnden Kuhhirten-Puppe Jessica Jane „Jessie“ Pride)
• Debian 9 *Stretch* (17. Juni 2017; benannt nach dem lila Kraken)
• Debian 10 *Buster* (6. Juli 2019; benannt nach dem Welpen aus *Toy Story 2*)
• Debian 11 *Bullseye* (14. August 2021; benannt nach dem Pferd von *Woody Pride*)
• Debian 12 *Bookworm* (Sommer 2023; benannt nach dem intelligenten Bücherwurm, einem Spielzeug mit eingebauter Leuchte aus *Toy Story 3*)

Es stehen bereits ebenfalls die Namen von zwei zukünftigen Veröffentlichungen fest:

• Debian 13 *Trixie* — benannt nach dem blauen Dinosaurier
• Debian 14 *Forky* — benannt nach dem aus Müll zusammengebauten Hauptcharakter aus *Toy Story 4*.


![Abbildung 2.9: Beschreibung der Filmserie Toy Story im Disney Wiki](image)
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. 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 übertragen. Dieses Archiv beinhaltet alle nicht mehr unterstützten Veröffentlichungen.


Experimental trägt – analog zu unstable – immer den Alias-Namen rc-buggy, was im Debian-Jargon eine Kurzform für „contains release-critical 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 Debianpaketes 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 veröffentlichtskritischen 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 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.


Auf diese Weise wird versucht, sämtliche veröffentlichtskritischen 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


### 2.10.6 Sicherheitsaktualisierungen


### 2.10.7 Long Term Support (LTS)


### 2.10.8 Extended Long Term Support (ELTS)


Für welche Pakete es Aktualisierungen gibt, hängt jedoch davon ab, ob ein Paket jemandem wichtig genug ist, um sich am Arbeitsaufwand für dessen Sicherheitsaktualisierungen zu beteiligen. Interessieren sich mehrere Personen oder Organisationen
für die Sicherheitsaktualisierungen desselben Paketes, so werden die Kosten entsprechend aufgeteilt. Die Koordination erfolgt über die französische Firma Freexian [Freexian].

Desweiteren gibt es im Vergleich zu LTS weitere Einschränkungen:

- Es werden nur Pakete unterstützt, für die sich Sponsoren finden. Die aktuelle Liste unterstützter Pakete findet sich unter [Debian-ELTS-Packages].
- Es werden ggf. noch weniger Architekturen unterstützt. Im Falle von Debian 8 Jessie sind dies nur noch i386, amd64 und armel.
- Der Linux-Kernel wird ggf. nicht unterstützt. Es wird jedoch ein Backport des Kernels von der darauffolgenden stabilen Debian-Veröffentlichung (die dann typischerweise zu diesem Zeitpunkt bereits unter Debian LTS gepflegt wird) angeboten. Im Falle von Debian 8 Jessie ist dies der Linux-Kernel 4.9 aus Debian 9 Stretch.


Um die von der erweiterten Langzeitunterstützung bereitgestellten Paketaktualisierungen nutzen zu können, müssen Sie im Gegensatz zu Debian LTS zwei Dinge tun — 1.) ein weiteres APT-Repository zu Ihrer /etc/apt/sources.list (oder einer Datei im Verzeichnis /etc/apt/sources.list.d/) hinzufügen, und 2.) den PGP-Schlüssel des Extended-LTS-Projektes importieren. Wie das erfolgt, ist im 'Debian ELTS-HowTo [Debian-ELTS-HowTo] beschrieben. Im Folgenden dazu eine kurze Zusammenfassung:


Danach wird das heruntergeladene Paket mit Administrator-Rechten (d.h. als root oder z.B. mittels sudo) über den Aufruf dpkg -i freexian-archive-keyring*.deb installiert. Nun wird das APT-Repository durch das Hinzufügen der folgenden Zeile aktiviert:

```
# sources.list-Eintrag für Extended LTS

# Generisch (passende Veröffentlichung und Archiv-Bereiche anpassen)
deb http://deb.freexian.com/extended-lts veröffentlichung-lts sektionen

# Beispiel für Debian 8 Jessie mit allen Archiv-Bereichen
deb http://deb.freexian.com/extended-lts jessie-lts main contrib non-free

# Beispiel für Debian 9 Stretch mit allen Archiv-Bereichen
deb http://deb.freexian.com/extended-lts stretch-lts main contrib non-free
```

Abschließend ist noch apt update oder ein Äquivalent aufzurufen, um die ELTS-Paketlisten herunterzuladen. Sind bereits Aktualisierungen verfügbar, so kann man diese direkt auch mit apt upgrade oder ggf. apt full-upgrade einspielen.

### 2.11 Benennung einer Paketdatei


2.11.1 Paketname

*Feld 1* bezeichnet den Namen des Pakets, welches durch die Paketdatei bereitgestellt wird. Die Paketdatei *iceweasel_3.5.16-12_i386.deb* ist ein Binärpaket (Dateiendung .deb) und beinhaltet den Webbrowser Iceweasel in der Version 3.5.16 für die Architektur i386.

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.


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 Paket *libxslt1-dev* befinden sich beispielsweise 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


Handelt es sich um ein *nicht-natives Debian-Paket*, besteht die Versionsnummer aus der Upstream-Version und der Debian-Revision. Beim 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 Bindestreiche, ist immer der letzte Bindestreich 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 die 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.

+nmun>

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

~bpo<x>+<y>

+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 Telefoniessware 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

**Inhalt der Liste der Architekturen**

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


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 8.43 nach.

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


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]
Fetched 6769 kB in 6s (1005 kB/s)
Reading package lists...  Done
```


Prüfung auf Verfügbarkeit für eine Architektur mittels apt-cache

```
# apt-cache policy pforth
pforth:i386:
   Installed: (none)
   Candidate: 21-12
   Version table:
```
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

```bash
# 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: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

```bash
# dpkg -l pforth libc6
```
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**

```bash
$ 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.

### 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 unbenützbar, da z.B. Pakete wie APT, less oder ein Texteditor fehlen. Die letztgenannten sind zum Betrieb nicht zwingend erforderlich1.

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 ein.

Als Pakete gehören neben apt u.a. gnupg und debian-archive-keyring für den Debian-Archiv-Schlüsselring zum Überprüfen der Signaturen von Paketlisten (siehe Abschnitt 8.32.1) 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 Bedienoberfläche. 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“-Modulen2), 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 „Bugreports anzeigen“ in Abschnitt 37.3).

2.13.4 Prioritätsstufe „optional“ (optional)


2.13.5 Prioritätsstufe „extra“ (extra)


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.


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

---


2.13.6 Markierung „essentiell“ (essential)


**Eintrag in der Paketbeschreibung**

| Essential: yes |


Unter „Pakete nach Prioritäten finden“ in Abschnitt 8.9 lesen Sie, wie Sie auflisten, welche Pakete genau auf Ihrer Version von Debian als essentiell markiert sind.


**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) sorgt dafür, dass eine möglichst hohe Softwarequalität in Debian gehalten wird. Neben den Werkzeugen zur Qualitätssicherung (siehe „Qualitätskontrolle“ in Kapitel 37) 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.


Danach wird in regelmäßigen Abständen — i.d.R. wöchentlich — der Softwarebestand (d.h. die installierten Pakete) erfasst, an das Popcon-Projekt übertragen und danach anonymisiert ausgewertet. Über die Projektwebseite erfolgt eine tabellarische Auswertung und eine graphische Auswertung. Abbildung 2.10 zeigt beispielhaft das Ergebnis für das Paket `nginx`.

Abbildung 2.10: 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 zugegriffen? Dies wird für die Graphen `vote` (dafür stimmen) und `old` (alt) verwendet und anhand der Zeitstempel des Zugriffs (`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

2.15 Lokale Paketmarkierungen


2.15.1 Paketmarkierungen, die von verschiedenen Programmen genutzt werden

Diese Markierungen werden teilweise bereits automatisch von APT und aptitude gesetzt, wenn es Pakete installiert, entfernt oder aktualisiert. Als Systembetreuer können Sie jederzeit eingreifen und die Markierungen eigenhändig setzen und entfernen. Die folgenden Paketmarkierungen sind in Benutzung:

automatisch installiert (automatic)

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 mehr von ihm abhängt (siehe „Umgang mit Waisen“ in Abschnitt 8.44).

halten (hold)
das Paket wird in der vorliegenden, installierten Version auf dem System gehalten und nicht aktualisiert (upgrade) oder deinstalliert (siehe „Pakete aktualisieren“ in Abschnitt 8.41 und „Pakete deinstallieren“ in Abschnitt 8.43).


Abbildung 2.11: Vergleich Installationen und Nutzung der Pakete screen und tmux

---

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

... 
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)  
Von Ihnen als lokaler Administrator nicht erwünschte Version, die nicht installiert wird, auf die nicht aktualisiert wird bzw. die beim Aktualisieren übersprungen wird.

Neue Pakete (New Packages/Unseen)  

Entfernungsgrund (Remove-Reason)  
aptitude zeigt an, warum ein Paket entfernt wird: wegen nicht (mehr) erfüllter Abhängigkeiten, wegen Konflikten mit anderen Paketen, oder weil es nicht mehr gebraucht wird (sprich: 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-spezifische Zusatzinformationen zu Paketen (Ausschnitt)

... 
Package: python3-pkg-resources  
Architecture: amd64  
Unseen: no
Diese benutzerspezifischen Markierungen werden auch in der Textoberfläche (text-based user interface, kurz TUI) von aptitude angezeigt, jedoch können Sie diese dort nicht ändern.

### 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 `python-pkg-resources`.

Auf der Kommandozeile rufen Sie hierfür aptitude mit dem Unterkommando `show` gefolgt vom Paketnamen auf. In den Zeilen 2 und 3 der nachfolgenden Ausgabe erfahren Sie einerseits, dass das Paket `python-pkg-resources` automatisch installiert wurde und die Version 18.8-1 nicht lokal eingespielt werden darf. Darüberhinaus wurde eine manuelle Markierung vergeben (`broken-by-807773`), die kennzeichnet, dass das Paket defekt ist (`broken`). Die Ziffernfolge referenziert die Nummer des Bugs im Debian Bug Tracking System (BTS) und ermöglicht Ihnen, nachzulesen, warum der Eintrag da ist.

#### 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 2.12).

![Abbildung 2.12: Ausgabe der Paketmarkierungen in der Textoberfläche von aptitude](attachment:image.png)

### 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 folgenden sechs Unterkommandos

- `showauto`
  - alle automatisch installierten Pakete

- `showmanual`
  - alle manuell installierten Pakete

- `showinstall`
  - alle Pakete, die zur Installation vorgemerkt sind

- `showhold`
  - alle Pakete, deren Zustand beibehalten wird

- `showremove`
  - alle Pakete, die zur Entfernung vorgemerkt sind

- `showpurge`
  - alle Pakete, die zur Entfernung inklusive der Konfiguration vorgemerkt sind


**Auflistung aller manuell installierten Pakete mittels `apt-mark`**

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

`apt-mark` erlaubt keine Eingrenzung, welche Pakete überprüft werden. Es validiert stets den gesamten Paketbestand.

### 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 diese sechs Schalter:

- `auto`
  - automatisch installiert

- `install`
  - Paket wird installiert

- `manual`
  - Paket wird manuell installiert
**hold**
Paket wird beibehalten

**purge**
Paket inklusive Konfiguration löschen

**remove**
Paket löschen


**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.

**Liste der Pakete eingrenzen, deren Markierung geändert wird**
Um nur eine Auswahl an Paketen zu markieren, erlaubt `apt-mark` eine Paketliste in Form einer Datei, die Sie beim Aufruf mit übergeben:

```
apt-mark -f=paketliste manual
```

Die Datei ist eine Textdatei, in der pro Zeile ein Paketname steht. Mit obigem Aufruf werden alle Pakete auf „manuell installiert“ gesetzt, die in der übermittelten Paketliste angegeben sind.

### 2.15.6 Was passiert, wenn Paketmarkierungen geändert werden?

Durch das Setzen von Paketmarkierungen verändert sich die Art und Weise, wie die Paketabhängigkeiten bewertet werden. `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.


### 2.15.7 Setzen und Entfernen einer Markierung mit `aptitude`

Alternativ zu `apt-mark` bietet sich auch `aptitude` an. Dort heißen die Unterkommandos etwas anders, ebenso agiert `aptitude` vielleicht ungewohnt. In der Standardeinstellung will es Pakete entfernen, die mangels geänderter Abhängigkeiten nicht mehr benötigt werden. 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.
```

#
Möchten Sie eine Markierung wieder aufheben, kennt aptitude den Schalter unmarkauto. Das nachfolgende Beispiel demonstriert das Vorgehen.

**Aufheben von Paketmarkierungen mit aptitude**

```bash
# 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.
```

Dabei fällt auf, dass aptitude im Gegensatz zu apt-mark nicht angibt, dass sich eine Markierung geändert oder nicht geändert hat. Stattdessen informiert es Sie darüber, dass es keine Pakete entfernt oder aktualisiert. Kurioserweise aktualisiert es (in der Standardeinstellung) nicht automatisch die Pakete, bei denen die hold-Markierung entfernt wurde:

**Setzen eines Paketes auf hold mit aptitude**

```bash
# aptitude search '~U'
iA 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.
```

```bash
# aptitude search '~U'
iA 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.
```

```bash
# aptitude search '~U'
iA awesome - Hochkonfigurierbarer Fenstermanager für X
```

2.16 Wie finde ich passende Pakete

### 2.16.1 Paketquellen


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 6.4.4) 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 „Pakete über den Namen finden“ in Abschnitt 8.20 genauer vor.

2.16.3 Paketeigenschaften und Einordnung

Teil II

Werkzeuge
Kapitel 3

Paketquellen und Werkzeuge

3.1 Paketquellen

3.1.1 Begriff und Hintergrund


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 von Ihnen 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 31).

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 von Ihnen oder einem Programm 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. Die Paketverwaltung erkennt die Änderungen nicht von sich aus und wartet auf ihren „Anstoß“. Danach synchronisiert sie die lokal vorliegenden Informationen über die verfügbaren Pakete und deren Abhängigkeiten (siehe Kapitel 7) mit den konfigurierten Paketquellen (Abschnitt 3.1).

Wir empfehlen Ihnen zur Aktualisierung den folgenden Ablauf:

1. Erstellen Sie zuerst eine Sicherheitskopie der entsprechenden Datei, z.B. `cp -pv /etc/apt/sources.list /etc/apt/sources.list.backup`. Gegebenenfalls macht das auch Ihr Texteditor automatisch.

2. Tragen Sie die neuen oder veränderten Paketquellen in die Datei `/etc/apt/sources.list` nach und speichern diese ab. Wenn Sie lediglich eine neue Paketquelle hinzufügen wollen, können Sie alternativ auch eine neue Datei mit dieser Paketquelle im Verzeichnis `/etc/apt/sources.list.d/` anlegen. Der Name dieser Datei muss dann auf `.list` enden, bspw. `skype.list` für die Paketquelle zum Kommunikationsprogramm Skype.


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. Seit 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 Die Datei `/etc/apt/sources.list` verstehen

3.3.1 Format der Paketliste

Wie auf UNIX/Linux-Systemen üblich, ist die Konfigurationsdatei `/etc/apt/sources.list` eine reine Textdatei. Die Einträge darin erfolgen zeilenweise. Jede einzelne Paketquelle beschreiben Sie vollständig in einer separaten Zeile.
Änderungen nehmen Sie mit Hilfe eines beliebigen Texteditors als Benutzer root vor, bspw. mittels *vim*, *emacs* oder *nano* (siehe Abbildung 3.1). Das Kommando *apt* verfügt zudem über ein Unterkommando namens *edit-sources*, welches Sie ebenfalls dahin bringt. Es nutzt den von Ihnen konfigurierten Texteditor. Haben Sie darin die Syntaxhervorhebung aktiviert, erfassen Sie die Struktur der einzelnen Einträge leichter und bemerken sofort fehlerhafte Änderungen durch eine entsprechende Einfärbung des Textes.


### 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*, *aptitude update* oder *apt update* (siehe Abschnitt 8.41) 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 künftig keine bösen Überraschungen mehr.


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

**Art der Quelle**


**URI**

legt die Art der Installationsquelle fest. Hierbei sind diese Angaben zulässig:

- **file**: die 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**: genutzt wird eine CD, eine DVD oder eine Blu-ray als Installationsmedium
- **http**: die Installationsquelle ist ein HTTP-Server
- **https**: die Installationsquelle ist ein HTTPS-Server
- **ftp**: die Installationsquelle ist ein FTP-Server
- **copy**: identisch zum Eintrag **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)

**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. **Bullseye**, **Bookworm** oder **Sid** (siehe Abschnitt 2.10.2).

Bitte beachten Sie bei Debian und Ubuntu die vollständige 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:

```plaintext
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.


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 **bookworm**:

```plaintext
deb http://ftp.de.debian.org/debian/ bookworm main contrib non-free non-free-firmware
```

In welcher Reihenfolge Sie die einzelnen, gewünschten Paketbereiche angeben, spielt keine Rolle. Üblich ist jedoch die Abfolge anhand des Freiheitsgrades der Softwarelizenz in der Form von **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 beeinflussen Sie die Bezugszeiten für Aktualisierungen der Paketlisten und der Pakete erheblich zu ihren Gunsten.

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
```


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

Automatisierung der Eintragung


3.3.5 Einträge für Sicherheitsaktualisierungen


Typischerweise wird dieser bereits zum Installationzeitpunkt vom Debian Installer angelegt, falls die entsprechende Frage mit "Ja" beantwortet haben.

Hatten Sie während der Installation bei der Frage nach Sicherheitsaktualisierungen "Nein" ausgewählt, oder fehlt der Eintrag aus sonstigen Gründen, so können Sie diesen manuell nachtragen.

Allerdings ist an dieser Stelle darauf zu achten, dass sich das Format des Eintrages zwischen Debian 10 Buster und Debian 11 Bullseye leicht geändert hat.

**sources.list-Eintrag für Sicherheitsaktualisierungen bis Debian 10**
```
deb http://security.debian.org/ <veröffentlichungsname>/updates <archivbereiche>
```

**sources.list-Eintrag für Sicherheitsaktualisierungen ab Debian 11**
```
deb http://security.debian.org/ <veröffentlichungsname>-security <archivbereiche>
```

Entsprechend hier Beispiele für Debian 10 Buster, Debian 11 Bullseye und Debian 12 Bookworm:

**sources.list-Eintrag für Sicherheitsaktualisierungen in Debian 10 Buster**
```
deb http://security.debian.org/ buster/updates main contrib non-free
```

**sources.list-Eintrag für Sicherheitsaktualisierungen in Debian 11 Bullseye**
```
deb http://security.debian.org/ bullseye/updates main contrib non-free
```

**sources.list-Eintrag für Sicherheitsaktualisierungen in Debian 12 Bookworm**
```
deb http://security.debian.org/ bookworm/updates main contrib non-free
```

### 3.3.6 Einträge für zusätzliche, nicht-offizielle Pakete


Im folgenden Beispiel sehen Sie die Einbindung der Paketquellen des PostgreSQL-Projekts [APT-Repo-PostgreSQL] und des X2Go-Projekts [APT-Repo-X2Go] für Debian 10 `Buster`:

```
deb https://apt.postgresql.org/pub/repos/apt/ buster-pgdg main
deb https://packages.x2go.org/debian buster 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 Verweis¹ auf den Bereich `non-free` auf dessen Paketserver:

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

#### Ergänzung der Signatur der Paketquelle


#### 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 12 `Bookworm`:

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

¹Die aktuelle Konfiguration des APT-Repositories erlaubt nur die Verwendung von `stable` als Veröffentlichung. Verwenden Sie z.B. `stretch` anstatt von `stable`, so beschwert sich APT, dass dies nicht vorgesehen sei.
3.3.8 Einträge für Deutschland

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

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

Mit den ersten beiden Zeilen beziehen Sie alle Binär- und Sourcepakete für die Distributionsbereiche `main`, `contrib` und `non-free` für die Veröffentlichung Debian 12 *Bookworm* vom primären deutschen Debian-Spiegelserver. Mit den Zeilen drei und vier beziehen Sie zusätzlich die zugehörigen Sicherheitsaktualisierungen für alle Distributionsbereiche der gleichen Veröffentlichung von der zentralen Stelle `security.debian.org`.


3.4 Geeigneten Paketmirror auswählen


Mit der oben beschriebenen, dezentralen Verteilung ist gewährleistet, dass Sie bei einem Ausfall oder der Nichtverfügbarkeit des von ihnen gewählten 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 Netzlücke auf unterschiedliche Mirrors und deren Standorte verteilt.

Für Sie bedeutet das im Alltag, 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 8.41.4).


3.4.1 Paketmirror bei Debian


*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
```

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].


**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/ |
| 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, bei Ubuntu hingegen im Entwicklerportal [Ubuntu-Mirrors]. In Abbildung 3.2 sehen Sie die Zusammenstellung für die Distribution Linux Mint.
### 3.4.3 Pakete ohne Paketmirror beziehen


### 3.5 Am besten erreichbaren Paketmirror finden


Über die Einträge in der Datei `/etc/apt/sources.list` (siehe Abschnitt 3.3) steuern Sie, welchen verfügbaren Paketmirror Sie benutzen, um den Softwarebestand auf ihrem System aktuell zu halten. Je leistungsfähiger der von Ihnen gewählte Paketmirror ist, umso weniger Zeit benötigen Sie im Endeffekt, um die lokale Aktualisierung vorzubereiten und durchzuführen.


### 3.5.1 netselect und netselect-apt

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
Debian-Paketmanagement

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.

Zwischen netselect und netselect-apt bestehen die folgenden Unterschiede:

- netselect gibt nur den ermittelten Zahlenwert für den evaluierten Spiegelserver aus.
- netselect-apt erzeugt eine Datei namens sources.list in dem Verzeichnis, in welchem Sie netselect-apt aufrufen. netselect-apt überschreibt dabei die Datei /etc/apt/sources.list nicht von sich aus. Die generierte Datei beinhaltet die besten gefundenen Spiegelserver und kann von Ihnen danach als neue Liste der Paketquellen benutzt werden. Dazu kopieren Sie die generierte Datei sources.list in das Verzeichnis /etc/apt/.
- netselect-apt ist nicht (mehr) für Ubuntu paketiert [netselect-apt-ubuntu]. Es steht für Debian zur Verfügung und funktioniert zuverlässig in allen Debian-Versionen.

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.

3.5.1.1 Paketquellen nach Pingzeiten und Entfernung auswählen


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.

Etwas ausführlichere Ausgabe 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.
.............................................................
128.61.240.89 141 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.


**Noch mehr Informationen zu den Paketmirrors**

```bash
# 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.debian.org  48 ms  14 hops - OK
128.30.2.36  119 ms  19 hops - OK
64.50.233.100 113 ms  12 hops - OK
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.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
```


### 3.5.1.2 Anzahl der HOPS begrenzen


**Paketmirror mit den wenigsten Zwischenknoten**

```
# netselect -m 10 -vvv ftp.at.debian.org
Running netselect to choose 1 out of 1 address.
ftp.at.debian.org      33 ms 5 hops  - HIGHER
ftp.at.debian.org      51 ms 8 hops  - HIGHER
ftp.at.debian.org      51 ms 9 hops  - HIGHER
ftp.at.debian.org      47 ms 10 hops - OK
min_lag is now 47
ftp.at.debian.org      49 ms 10 hops - OK
ftp.at.debian.org      48 ms 10 hops - OK
ftp.at.debian.org      56 ms 10 hops - OK
ftp.at.debian.org      49 ms 10 hops - OK
ftp.at.debian.org      48 ms 10 hops - OK
ftp.at.debian.org      48 ms 10 hops - OK
ftp.at.debian.org      48 ms 10 hops - OK
```
### 3.5.1.3 Einen geschützten Paketmirror abfragen

Ist der Paketmirror beispielsweise von einer Firewall geschützt und diese blockiert UDP-Pakete, kann die Option `-I` von größerem Nutzen sein. Damit sendet netselect zur Abfrage stattdessen ICMP-Pakete und umgeht das Hindernis. Das Ergebnis sehen Sie in der nachfolgenden Ausgabe:

**Paketmirror mit ICMP-Paketen abfragen**

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

### 3.5.1.4 Liste der Paketquellen mit netselect-apt generieren lassen

Wie bereits in Abschnitt 3.5.1 angesprochen, erzeugt netselect-apt eine Datei `sources.list` im aktuellen Verzeichnis. Es verfügt zudem über den Schalter `-o` (Langform `--outfile`), mit dem Sie die entsprechende Ausgabedatei angeben und eine passende Liste darin generieren lassen. netselect-apt akzeptiert für die Wahl der Veröffentlichung auch Angaben wie `stable` oder `unstable`, aber auch die Alternativnamen der Veröffentlichung wie `Bookworm` oder `Sid` (siehe „Veröffentlichungen“ in Abschnitt 2.10).

Insgesamt kennt netselect-apt diese Schalter zur Steuerung der Liste:

- `-a` (Langform `--arch`)

- `-c` (Langform `--country`)
  Die Einträge kommen nur aus dem angegebenen Land.

- `-f` (Langform `--ftp`)
  Benutze FTP-Quellen anstatt von HTTP-Quellen.
-n (Langform --nonfree)
  Ergänzung der Einträge um den Distributionsbereich nonfree (siehe „Distributionsbereiche“ in Abschnitt 2.9).

-o (Langform --outfile)
  Speichere die erzeugte Liste in der angegebenen Datei.

-s (Langform --sources)
  zusätzliche Erzeugung von Einträgen für den Bezug von Quellpaketen (siehe „Sourcepakete“ in Abschnitt 2.7.4 und „Einträge für Quellpakete“ in Abschnitt 3.3.7).

Im nachfolgenden Beispiel kommt zunächst lediglich der Schalter `-o test.list` zum Einsatz. `netselect-apt` erzeugt die Liste der ermittelten Paketmirrors in der Datei `test.list` im lokalen Verzeichnis.

**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...
```

```
Auflösen des Hostnamen »www.debian.org (www.debian.org)«... 5.153.231.4, 130.89.148.14, ...
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.WrClOIS« speichern.
100%[=================================================================] 338.381 959K/s ← in 0,3s
2014-02-13 14:55:03 (959 KB/s) - »/tmp/netselect-apt.WrClOIS« 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)...
... 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.lund1.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.
#`
```
Das zweite Beispiel kommt aus dem Alltag. Wir kombinieren die vier Schalter \(-c\), \(-t\), \(-n\) und \(-a\), um die besten fünf Paketmirror für die Architektur amd64 in Frankreich zu finden:

**Ermittlung der besten fünf Paketmirror**

```
# netselect-apt -c france -t 5 -a amd64 -n stable
Using distribution stable.
Retrieving the list of mirrors from www.debian.org...

Auflosen des Hostnamen »www.debian.org (www.debian.org)<... 130.89.148.14, 5.153.231.4, ...
Verbindungsaufbau zu www.debian.org (www.debian.org)|130.89.148.14|:80... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 302 Found
Platz: https://www.debian.org/mirror/mirrors_full[folge]
Verbindungsaufbau zu www.debian.org (www.debian.org)|130.89.148.14|:443... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 189770 (185K) [text/html]
In »/tmp/netselect-apt.Kp2SNk« speichern.
```

```
/tmp/netselect-apt.Kp2SNk 100%[================================================================]=>
185,32K 1,19MB/s in 0,2s
```

```
2019-01-09 11:47:22 (1,19 MB/s) - »/tmp/netselect-apt.Kp2SNk« gespeichert [189770/189770]
```

```
Choosing a main Debian mirror using netselect.
(will filter only for mirrors in country france)
netselect: 19 (19 active) nameserver request(s)...
Duplicate address 212.27.32.66 (http://debian.proxad.net/debian/, http://ftp.fr.debian.org/)
keeping only under first name.
Running netselect to choose 5 out of 18 addresses.
```

```
The fastest 5 servers seem to be:
```
  http://debian.proxad.net/debian/
  http://debian.mirror.ate.info/
  http://debian.mirrors.ovh.net/debian/
  http://ftp.rezopole.net/debian/
  http://mirror.plusserver.com/debian/debian/
```

```
Of the hosts tested we choose the fastest valid for HTTP:
http://debian.proxad.net/debian/
```

```
Writing sources.list.
Done.
```
```
```
```
```
```
```
```

Die von netselect-apt erzeugte Datei 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**

```
# 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-security main contrib
```
3.5.1.5 *netselect* und *netselect-apt* im Alltagseinsatz

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 interesserhalber 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.

### 3.6 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.


**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
```


#### 3.6.2 Paketquellen über GeoIP auswählen


Diese Funktionalität 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 Per CDN


Die Sicherheitsaktualisierungen von Debian kommen 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 schon seit längerem ein CDN.

Seit einigen Jahren ist aber auch die Standard-Einstellung in der `sources.list`-Datei von Debian der Hostname `deb.debian.org` der auch wiederum auf ein CDN zeigt, zur Zeit auf das CDN von Fastly [debian-partners-fastly].

Eine DNS-Anfrage auf diesen Hostnamen aus der Schweiz sieht so aus:

```bash
$ host deb.debian.org
deb.debian.org is an alias for debian.map.fastlydns.net.
debian.map.fastlydns.net has address 146.75.122.132
debian.map.fastlydns.net has IPv6 address 2a04:4e42:8e::644
```

Eine DNS-Anfrage aus Schweden dagegen anders:

```bash
$ host deb.debian.org
deb.debian.org is an alias for debian.map.fastlydns.net.
debian.map.fastlydns.net has address 199.232.18.132
debian.map.fastlydns.net has IPv6 address 2a04:4e42:41::644
```
3.6.4 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-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
```


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.2


```
deb mirror://mirrors.ubuntu.com/DE.txt trusty main contrib non-free
```

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

Unter `https://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
```

---

2 Um keine unübersichtlich langen Beispiele abzudrucken, wurden hier absichtlich die beiden Beispiele aus dem deutschsprachigen Raum gewählt, die relativ kurze Listen ergeben.
3.6.5 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.

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'
...
  netselect/netselect_0.3.dsn-25_amd64.deb%0aFilename:%20/var/cache/apt/archives/partial
  netselect_0.3.dsn-25_amd64.deb%0aMessage:%20Connecting%20to%20ftp.ch.debian.org
...
```


**Demonstration**

Your details, as seen by the redirector:

- IP: 217.92.222.228
- AS: 3220
- Continent: EU
- Country: DE
- Region: null
- City: null

Had you requested a file from debian, you would have been sent to one of the following mirrors:

- http://mirror.ubuntu.de/debian/
- http://debian.mirror.info/debian/
- http://debian.netcologne.de/debian/
- http://mirror.de.debian.web.net/debian/

Out of a population of 11 mirrors
  Matched by country

Abbildung 3.4: 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?


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 3.5: 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` 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.


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 Schalter und Aufrufe:
### Identifikation eines Installationsmediums

```plaintext
# apt-cdrom ident
Verwendeter CD-ROM-Einbindungspunkt: /media/cdrom/
CD-ROM wird eingebunden.
Identifizieren ... [3e81e0fb1b74074c6e427e18afe3ab7-2]
Gespeicherte Kennzeichnung:
Einbindung der CD-ROM wird gelöst ...
```

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


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):

```plaintext
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


Über die verschiedenen Reiter stellen Sie die gewünschten Paketquellen ein. Abbildung 3.6 zeigt die Einstellungen zu den Standard-Debian-Repositories.

![Abbildung 3.6: Einstellung der Komponenten in Synaptic](image)

3.11 Debian und Ubuntu Sources List Generator

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 3.7). Über den Knopf Generate sources list erstellt Ihnen das Programm eine passende Liste der Paketquellen (siehe Abbildung 3.8). Diese Datei können Sie nun als neue /etc/apt/sources.list in Ihr System übernehmen.

Abbildung 3.7: Auswahl der Komponenten im Debian Sources List Generator

Abbildung 3.8: 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 3.9: Auswahl der Komponenten im Ubuntu Sources List Generator

#### 3.12 Paketquelle auf Echtheit überprüfen

##### 3.12.1 Basiswissen


Hintergrund dafür ist einesseits, dass eine Paketquelle Paketarchive unterschiedlichster Herkunft umfasst. Die Daten könnten aus einer wenig vertrauenswürdigen Quelle stammen 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).


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)


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.


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.


![](image)

Abbildung 3.10: Auflistung der gespeicherten, vertrauenswürdigen Schlüssel

Mit dem Aufruf apt-key finger zeigen Sie zusätzlich deren Fingerausdruck an³. Nachfolgend sehen Sie beispielhaft die Signatures zum Opera Software Archive, dem Mendeley Desktop Team und dem Debian Archive für die beiden Veröffentlichungen Wheezy und Jessie.

³Da 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.
Liste der Signaturen (Ausschnitt)

```
# apt-key finger
/etc/apt/trusted.gpg

-- -------------------
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 D6FD 518E 17E1
uid Jessie Stable Release Key <debian-release@lists.debian.org>

/etc/apt/trusted.gpg.d/debian-archive-wheezy-automatic.gpg

-- ----------------------------------------------------------
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>
```

Mit dem Aufruf `apt-key export Schlüssel` geben Sie hingegen nur einen bestimmten Schlüssel auf der Standardausgabe als PGP-Block aus. Der Schalter `apt-key exportall` führt das gleiche für alle Schlüssel durch.

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.

**Anmerkung**

Ab Debian 9 *Stretch* ist diese Option als veraltet markiert.

### 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 Herstellerseite 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
Auflösen des Hostnamen »deb.opera.com (deb.opera.com)«... 185.26.183.130
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]
#

<table>
<thead>
<tr>
<th>Unverschlüsselte Übertragung von Schlüsseln</th>
</tr>
</thead>
<tbody>
<tr>
<td>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 allzu vielen Signaturen ausgestattet zu sein, sodass eine Verifizierung über die Signaturen ebenfalls nicht möglich ist.</td>
</tr>
</tbody>
</table>

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`

```bash
# 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.


#### 3.12.5 Abkündigung von `apt-key`


#### 3.12.6 Alternative Benutzerschnittstellen zur APT-Schlüsselverwaltung

Die Abkündigung von `apt-key` ist auch einer der Gründe, warum sich niemand mehr darum gekümmert hat. `gui-apt-key` [Debian-Paket-gui-apt-key], das verwaiste GUI-Frontend zu `apt-key`, weiterzuentwickeln. Entsprechend ist auch die darauf aufbauende, curses-basierende TUI-Programm `curses-apt-key` [curses-apt-key] nicht mehr weiterentwickelt wird.

#### 3.13 Liste der verfügbaren Pakete aktualisieren

##### 3.13.1 Grundlegendes Vorgehen

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 statt dessen zunächst File -> Update selected channels ... aus und entscheiden danach, welche Paketquelle Ihres Erachtens eine Auffrischung verdient hat (siehe dazu Abbildung 3.11).

Abbildung 3.11: Auflistung der verfügbaren Paketquellen in SmartPM

**Aktualisierung mit dpkg**

Führen Sie eines der o.g. Aufrufe aus, wird zunächst die Liste der Paketquellen in der Datei `/etc/apt/sources.list` und dem Verzeichnis `/etc/apt/sources.list.d` (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.

• 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
Holen: 1 http://security.debian.org wheezy/updates Release.gpg [836 B]
Holen: 2 http://ftp.de.debian.org wheezy/updates Release [102 kB]
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/updates/main Sources
OK http://ftp.de.debian.org wheezy/updates/main Sources [150 kB]
OK http://ftp.de.debian.org wheezy/updates/main Sources [14 B]
OK http://ftp.de.debian.org wheezy/updates/main Sources
OK http://ftp.de.debian.org wheezy/main i386 Packages
Holen: 6 http://security.debian.org wheezy/updates/main i386 Packages [150 kB]
OK http://ftp.de.debian.org wheezy/main Translation-de
OK http://ftp.de.debian.org wheezy/main Translation-de
Holen: 7 http://security.debian.org wheezy/updates/main i386 Packages [14 B]
Holen: 8 http://security.debian.org wheezy/updates/main i386 Packages [14 B]
Holen: 9 http://security.debian.org wheezy/updates/main i386 Packages [14 B]
Holen: 10 http://security.debian.org wheezy/main Translation-en [88,7 kB]
Holen: 11 http://security.debian.org wheezy/updates/main i386 Packages
Es wurden 421 kB in 0 s geholt (428 kB/s).
Paketlisten wurden gelesen... Fertig
```

Abbildung 3.12: Auflistung der aktualisierbaren Pakete

Für diese Mitteilungen greifen `apt-get` und `apt` auf das Werkzeug `daptup` aus dem gleichnamigen Paket zurück [Debian-Paket-daptup]. Es ist als eine direkte Abhängigkeit zu beiden definiert und wird daher automatisch installiert.

### 3.13.2 Überprüfung der Paketsignaturen

Konnten bei der Aktualisierung für neue Paketlisten keine gültigen Signaturen gefunden werden, wird eine Warnung ausgegeben. Entsprechende Zeilen beginnen mit `W:`. Bei einer Paketquelle ohne Schlüssel beschwert sich APT wie folgt:

**Aktualisierung der Paketlisten ohne passenden GPG-Schlüssel**

```bash
# 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
NO_PUBKEY E585066A30C18A2B
```

Pakete, die nicht korrekt signiert sind, können Schadcode enthalten und sollten nicht installiert werden. `aptitude` warnt Sie in diesem Fall sehr deutlich:
3.13.3 Platz für den Paketcache

Bitte planen Sie freien Platz für den Paketcache ein. Die aktualisierten Paketlisten und Pakete benötigen Speicherplatz, bevor diese ausgepackt und eingerichtet werden können.

3.13.4 Die Veröffentlichung wechseln


3.14 Lokale Paketliste und Paketcache


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.ch.debian.org und die Distributionsbereiche main, contrib und non-free im der Veröffentlichung buster benutzen und zusätzlich auch deb-src-Zeilen in der sources.list
haben (deswegen die Dateien mit `Sources` im Namen) und `apt-file` installiert haben (deswegen die Dateien mit `Contents` im Namen).

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

```bash
$ ls -F /var/lib/apt/lists
auxfiles/
  ftp.ch.debian.org_debian_dists_buster_contrib_binary-amd64_Packages
  ftp.ch.debian.org_debian_dists_buster_contrib_Contents-amd64.lz4
  ftp.ch.debian.org_debian_dists_buster_contrib_i18n_Translation-de
  ftp.ch.debian.org_debian_dists_buster_contrib_source_Sources
  ftp.ch.debian.org_debian_dists_buster_InRelease
  ftp.ch.debian.org_debian_dists_buster_main_binary-amd64_Packages
  ftp.ch.debian.org_debian_dists_buster_main_Contents-amd64.lz4
  ftp.ch.debian.org_debian_dists_buster_main_i18n_Translation-de
  ftp.ch.debian.org_debian_dists_buster_main_source_Sources
  ftp.ch.debian.org_debian_dists_buster_non-free_binary-amd64_Packages
  ftp.ch.debian.org_debian_dists_buster_non-free_Contents-amd64.lz4
  ftp.ch.debian.org_debian_dists_buster_non-free_i18n_Translation-de
  ftp.ch.debian.org_debian_dists_buster_non-free_source_Sources
lock
partial/
  security.debian.org_dists_buster_updates_InRelease
  security.debian.org_dists_buster_updates_main_binary-amd64_Packages
  security.debian.org_dists_buster_updates_main_i18n_Translation-de
  security.debian.org_dists_buster_updates_main_source_Sources
  security.debian.org_dists_buster_updates_non-free_binary-amd64_Packages
  security.debian.org_dists_buster_updates_non-free_Contents-amd64.lz4
  security.debian.org_dists_buster_updates_non-free_i18n_Translation-de
  security.debian.org_dists_buster_updates_non-free_source_Sources
```

Für jede Paketquelle aus `/etc/apt/sources.list` wird eine oder mehrere 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 `0ad-data` aus der Paketkategorie Spiele (`games`).

Eintrag in `/var/lib/apt/lists/debian.ethz.ch_debian_dists_bullseye_main_binary-amd64_Packages zum Paket 0ad-data

```
Package: 0ad-data
Version: 0.0.23.1-1.1
Installed-Size: 2044173
Maintainer: Debian Games Team <pkg-games-devel@lists.alioth.debian.org>
Architecture: all
Pre-Depends: dpkg (>= 1.15.6-)
Suggests: Oad
Description: Real-time strategy game of ancient warfare (data files)
Homepage: http://play0ad.com/
Description-md5: 26581e685027d5ae84824362a4ba59ee
Tag: role::app-data
Section: games
Priority: optional
Filename: pool/main/0/0ad-data/0ad-data_0.0.23.1-1.1_all.deb
Size: 70183824
MD5sum: b2b6e5510898abf00ee79da48995f92f
SHA256: afb3f0ddaceb36dc2d71ed83d7fede4ada419511a948ee4a06fa4dabc1b486e2c0
```

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

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 aus einem anderen Grund fehlschlägt. Sie bekommen das mit, wenn APT jammert und in Folge seine Arbeit verweigert.


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


#### 3.15.1 Aktualität des Mirrors überprüfen

Abbildung 3.14: Status der verschiedenen Debian-Paketmirror


---

4Bei 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.


\begin{table}[h]
\centering
\begin{tabular}{|c|l|l|}
\hline
\textbf{Operator} & \textbf{Beschreibung} & \textbf{Beispiel} \\
\hline
\texttt{<<} & früher als & \texttt{xpdf-utils \ (<} 3.00) \\
\texttt{<=} & früher oder gleich & \texttt{python-cairo \ (<=} 1.85) \\
\texttt{=} & exakt gleich & \texttt{xfwm4 \ (=} 4.1) \\
\texttt{>=} & gleich oder später & \texttt{libc6 \ (>=} 2.4) \\
\texttt{>>} & später als & \texttt{libaal \ (>>} 1.4) \\
\hline
\end{tabular}
\caption{Relationen für versionierte Abhängigkeiten}
\end{table}

4.1.1 Binärpakete

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

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

\textbf{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 8.22)

**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 Installationsskripten 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**

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

```plaintext
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~)
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 Schlüsselwörtern 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 Quellpaket, 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**

**Vcs-Git, Vcs-Svn, Vcs-Hg, Vcs-Cvs, Vcs-Mtn**
zu dt.: Versionskontrollsystem; Angabe, von wo Sie eine aktuelle Entwicklungskopie des Quellpaket 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 autopkgtetest (siehe Debian Enhancement Proposal DEP 8 [DEP-8] und das gleichnamige Debianpaket dazu [Debian-Paket-autopkgtest]).
### 4.1.3 Weitere Metadaten


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

<table>
<thead>
<tr>
<th>Metadaten</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description-md5:</td>
</tr>
<tr>
<td>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</td>
</tr>
<tr>
<td>Filename:</td>
</tr>
<tr>
<td>Size: 162034</td>
</tr>
<tr>
<td>MD5sum:</td>
</tr>
<tr>
<td>SHA1:</td>
</tr>
<tr>
<td>SHA256:</td>
</tr>
</tbody>
</table>

### 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.


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, lzo und lzma, wobei lediglich gzip, bzip2, lzo und lzma von Debian GNU/Linux unterstützt werden, während die anderen Tar-Archive mit folgender Bemerkung unterstützt werden, aber lediglich 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 *native* Paketen gesprochen.

Upstream-Quellcode


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

4.2.3.1 Komponenten


Auspacken von Paketen mit ar

```sh
$ ar -t autotools-dev_20100122.1_all.deb
debian-binary
ccontrol.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:

```sh
$ ar -t mplayer_2%3a1.0~rc4.dfsg1+svn34540-1+b2_i386.deb
debian-binary
ccontrol.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

4.2.3.2 Benennung

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

4.2.3.3 Steuerdateien und Skripten

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

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 8.32.1).

shlibs
Diese Datei listet Bibliotheken und Shared Object Name (kurz SONAME) auf, welches das Paket gemeinsam mit dem Paketnamen zur Verfügung stellt.

config

templates
Diese Datei enthält Texte zu den Fragen und Hinweisen, die debconf während der Paketkonfiguration anzeigt (siehe dazu auch „Pakete konfigurieren“ in Abschnitt 8.40).

4.2.3.4 Daten im Paket


Diese Struktur, die zu installierenden Dateien sowie deren Typ und Größe zeigen Sie mit dem Kommando dpkg-deb --c Paketdatei an. Das nachfolgende Beispiel anhand des Pakets vnstat zeigt, dass darin 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/
-rw-r-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/
-rw-r-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/
```
Wünschen Sie stattdessen eine graphische oder webbasierte Darstellung des Paketinhalts, stehen Ihnen als Alternativen die Werkzeuge *deb-gview*, *Synaptic*, *dpkg-www* und *apt-browse* zur Verfügung. Im Detail agiert hier jedes der genannten Programme anders.


*deb-gview* finden Sie im gleichnamigen Paket [ Debian-Paket-deb-gview ]. Abbildung 4.2 zeigt die Bedienoberfläche beispielhaft anhand des Pakets *debsums* [Debian-Paket-debsums]. Die dreispaltige Aufteilung beinhaltet die Daten- und Steuerdateien, die darin enthaltenen Programmdateien und Metadaten zum Paket.

Abbildung 4.3 zeigt die Programmdateien zum gleichen Paket, wie es Synaptic darstellt. Sie erreichen dieses Dialogfenster über *Paket* → *Eigenschaften* und danach im Reiter „Installierte Dateien“. Ausführlicher besprechen wir Synaptic in Abschnitt 6.4.1.
Abbildung 4.3: Detailinformationen zum Paket *debsums* (Synaptic)


Abbildung 4.4: Detailinformationen zum Paket *debsums* in apt-browser.org (Ausschnitt)

### 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 liegen in der Beschreibung der Abhängigkeiten der Pakete.


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 (Symbol) von `/bin/ash` zu `/bin/dash`:

**Symbolischer Verweis auf eine andere Komponente am Beispiel der ash**

```bash
$ 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-pkgX.Y (X.Y ist in Debian 9 Stretch und Debian 10 Buster 5.0, siehe [Debian-Paket-libapt-pkg5.0], und bei Debian 11 Bullseye und Debian 12 Bookworm ist es 6.0) 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 und rsh.

5.2 Bibliothek libapt-pkg-perl


5.3 Bibliothek python-apt

Diese Bibliothek aus dem Paket python-apt [Debian-Paket-python-apt] beinhaltet die Python-Schnittstelle zum Zugriff auf die einzelnen Softwarepakete. Es hat die gleiche Funktionalität wie die weiter oben beschriebenen Pakete libapt-pkg und libapt-pkg-perl.
5.4 Paket libapt-pkg-doc


5.5 Bibliothek libapt-inst

Um Informationen aus deb-Paketen zu erhalten, nutzen Sie diese Bibliothek aus dem Paket libapt-instX.Y (X,Y ist in Debian 9 Stretch und Debian 10 Buster 2.0, siehe [Debian-Paket-libapt-inst2.0]). 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).

Seit APT 1.9.0, Debian 11 Bullseye und Ubuntu 19.10 Eoan ist libapt-inst in libapt-pkg aufgegangen.
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 Zwischen und stellen mehrere Bedienmodi zur Verfügung, so bspw. SmartPM (Abschnitt 6.4.3), welches Sie sowohl über die Kommandozeile, als auch über eine grafische Oberfläche (GUI) bedienen können. aptitude (Abschnitt 6.3.2), cupi (Abschnitt 6.2.5) und wajig (Abschnitt 8.44.6) 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

<table>
<thead>
<tr>
<th>Kategorie</th>
<th>Debian-basierte Systeme</th>
<th>RPM-basierte Systeme</th>
<th>andere Paketformate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kommandozeile</td>
<td>dpkg, dpkg-www, APT,</td>
<td>rpm, yum, dnf, urpmi,</td>
<td>emerge, pacman,</td>
</tr>
<tr>
<td></td>
<td>aptitude, cupi,</td>
<td>zypper, SmartPM,</td>
<td>sysget</td>
</tr>
<tr>
<td></td>
<td>SmartPM, gdebi</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(gdebi-core), wajig,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>gdebi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text User Interface (TUI)</td>
<td>tasksel, aptitude,</td>
<td>Yet another Setup Tool</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Debian Installer,</td>
<td>(YaST), DrakConf oder</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Univention Installer</td>
<td>Mandriva Linux Control</td>
<td></td>
</tr>
<tr>
<td></td>
<td>für Univention</td>
<td>Center (MCC) [Mandriva-Wiki]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Corporate Server (UCS)</td>
<td>bzw. Mageia Control Center (MCC) (Textkonsole)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pcurses</td>
</tr>
<tr>
<td>Graphical User Interface</td>
<td>Synaptic, SmartPM,</td>
<td>Yet another Setup Tool</td>
<td>PacmanXG4,</td>
</tr>
<tr>
<td>(GUI)</td>
<td>Muon, PackageKit,</td>
<td>2 (YaST2), DrakConf oder</td>
<td>PacmanExpress,</td>
</tr>
<tr>
<td></td>
<td>Apper (früher</td>
<td>Mandriva Linux Control</td>
<td>tkPacman, GNOME</td>
</tr>
<tr>
<td></td>
<td>KPackageKit), gdebi</td>
<td>Center (MCC) [Mandriva-Wiki]</td>
<td>PackageKit,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bzw. Mageia Control Center (MCC)</td>
<td>Zenity Pacman GUI, Octopi</td>
</tr>
</tbody>
</table>
Tabelle 6.1: (continued)

<table>
<thead>
<tr>
<th>Kategorie</th>
<th>deb-basierte Systeme</th>
<th>rpm-basierte Systeme</th>
<th>andere Paketformate</th>
</tr>
</thead>
<tbody>
<tr>
<td>webbasierte Verwaltung  (WUI)</td>
<td>IP Brick [ipbrick], Univention Management Console für Univention Corporate Server (UCS), Ubuntu Landscape , Appnr, Communtu, Debian Pure Blends</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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 Paketdiateien 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 8.38).


### 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 49.2).


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 B) 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 sind1:

- Paketliste ausgeben mittels dpkg -l (siehe Abschnitt 8.5)
- Paketstatus erfragen mit dpkg -s Paketname (siehe Abschnitt 8.4)
- Inhalt eines installierten Pakets anzeigen mit dpkg -L Paketname (siehe Abschnitt 8.26)
- Inhalt eines nicht installierten Pakets anzeigen (siehe Abschnitt 8.26) mit dpkg -c Paketname
- Paket zu Datei finden (siehe Abschnitt 8.24) mit dpkg -S Dateiname und
- Pakete konfigurieren (siehe Abschnitt 8.40) (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

6.2.2.1 Überblick

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 Ausführungs geschwindigkeit 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. Es 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).

1Weitere Optionen zu dpkg entnehmen Sie bitte der Manpage zum Programm
6.2.2.2 Komponenten und Funktionen

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 8.26) mit apt-file show Paketname
- Paketabhängigkeiten anzeigen (Abschnitt 8.19) mit apt-cache depends Paketname
- Paket über den Namen oder die Beschreibung finden (Abschnitt 8.20) mit apt-cache search Paketname
- Paket installieren (Abschnitt 8.38) mit apt-get install Paketname
- Installierte Pakete löschen (Abschnitt 8.43) mit apt-get remove Paketname
- Paketliste aktualisieren (Abschnitt 3.13) mit apt-get update
- neue Versionen für die Pakete einspielen (Abschnitt 8.41) mit apt-get upgrade
- die gesamte Distribution aktualisieren (Abschnitt 8.47) 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.

6.2.2.3 apt-cache

apt-cache bietet die folgenden Unterkommandos:

- depends
  Paketabhängigkeiten anzeigen (siehe Abschnitt 8.19)
- 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 8.14 und Abschnitt 8.14.3)
- pkgnames
  die Namen aller Pakete auflisten, die APT kennt (siehe Abschnitt 8.3)
- policy
  die Quellen und deren Prioritäten auflisten (siehe Abschnitt 8.14)
- rdepends
  umgekehrte Paketabhängigkeiten anzeigen (siehe Abschnitt 8.19)
- search
  Paket über den Namen finden (siehe Abschnitt 8.20)
show
Paketinformationen ausgeben und Paketstatus erfragen (siehe Abschnitt 8.4)

showsrc
Informationen zum Sourcepaket anzeigen (siehe Abschnitt 8.37)

showpkg
Informationen über das Paket anzeigen (siehe Abschnitt 8.4)

stats
Statistik zum Paketcache ausgeben (siehe Abschnitt 7.3)

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

### 6.2.2.4 apt-get

apt-get gehört mit Sicherheit zur Menge der gebräuchlichsten Kommandos der APT-Familie und verfügt über die folgenden Unterkommandos:

autoclean
Paketcache aufräumen (siehe Abschnitt 7.5)

autoremove
Paketwaisen löschen (siehe Abschnitt 8.44)

build-dep
Abhängigkeiten eines Sourcepakets erfüllen (findet Verwendung beim Erstellen von Paketen)

check
Paketcache auf beschädigte Paketabhängigkeiten prüfen (siehe Abschnitt 8.19)

clean
Paketcache aufräumen (siehe Abschnitt 7.5)

dist-upgrade
Distribution aktualisieren (siehe Abschnitt 8.47)

download
Paketdatei nur herunterladen (siehe Abschnitt 8.34)

dselect-upgrade
Aktualisierung der Pakete über dselect

install
Paket installieren (siehe Abschnitt 8.38)

purge
Paket inklusive Konfigurationsdateien des Pakets entfernen (siehe Abschnitt 8.43)

remove
Paket deinstallieren (siehe Abschnitt 8.43)

source
Beziehen der Sourcepakete (siehe Abschnitt 8.36)

update
Paketliste aktualisieren (siehe Abschnitt 3.13)

upgrade
Pakete auf eine neue Version aktualisieren (siehe Abschnitt 8.41)
6.2.2.5 apt-key und apt-mark


6.2.2.6 Weiterentwicklung von APT


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 Programname 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 8.41 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

In LinuxMint gibt es dagegen schon länger einen Befehl apt [LinuxMint-apt], welcher allerdings ein in Python geschriebener Wrapper um apt-get, apt-cache und neuerdings auch apt ist. Dieser befindet sich in /usr/local/bin/ und hat weitere LinuxMint-spezifische Features, wie z.B. das automatische Aufrufen der eigentlichen Befehle via sudo wo notwendig.

Ebenfalls in produktivem Zustand und teilweise intensiver Benutzung befinden sich die Werkzeuge cup, aptitude und SmartPM. Während sich und cup nur auf die Kommandozeile beschränken, bieten Ihnen aptitude zusätzlich eine text-basierte bzw. SmartPM eine graphische Benutzeroberfläche. Auf diese Werkzeuge gehen wir nachfolgend genauer ein (siehe, Abschnitt 6.2.5, Abschnitt 6.3.2 und Abschnitt 6.4.3).


6.2.3 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 nicht mehr verfügbare aptsh, beinhaltet aber auch Elemente von cup (Abschnitt 6.2.5) und aptitude (Abschnitt 6.3.2) auf der Kommandozeile 3.

**Anmerkung**

Die bisher letzte stabile Veröffentlichung von wajig befindet sich in Debian 10 Buster, danach ist es nur noch im Bereich unstable vorrätig.

wajig zielt darauf ab, alle im Alltag erforderlichen Aktionen zur Paketverwaltung in einem einzigen Werkzeug für die Kommandozeile zusammenzufassen. Daher haben sich die wajig-Entwickler das Ziel gesetzt, die APT-Bibliotheken (siehe Kapitel 5) vollständig auszurüsten 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 23.2) entlehnt wurden.

---

2Diese Eigenschaft stammt vom Programm gdebi (siehe Abschnitt 6.4.5), welches ebenfalls vom APT-Entwickler Michael Vogt gepflegt wird.

3Bis einschließlich Debian 6 Squeeze bestand zudem eine graphische Variante namens gjig, die mittlerweile obsolet und in keiner unterstützten Debian- oder Ubuntu-Veröffentlichung mehr verfügbar ist.
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 8.44) anzeigt.


**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/doc/apt/de/man5/sources.list.5.gz
debtags: /etc/debtags/sources.list.d/source-example
apt: /usr/share/doc/apt/man5/sources.list.5.gz
apt: /etc/apt/sources.list.d
apt: /usr/share/man/man5/sources.list.5.gz
```

Abbildung 6.1: wajig mit der Ausgabe des Kommandos listfiles


### 6.2.4 sysget

**sysget** ist ein Wrapper, welches den Aufruf zu den verschiedenen, tatsächlichen Paketwerkzeugen übernimmt. Ziel ist dabei, eine einheitliche Schnittstelle zu den Programmen wie APT, yum oder DNF herzustellen und damit auch Einsteigern die ersten Schritte nach einem Wechsel der Distribution zu erleichtern. **sysget** wird als Projekt auf GitHub gepflegt [sysgetGitHub].

Das Werkzeug orientiert sich an APT und versteht derzeit die folgenden Unterkommandos:
autoremove
Paketwaisen entfernen

clean
Aufräumen des Paketcaches

install
ein Paket installieren

remove
ein installiertes Paket wieder entfernen

search
Suche nach einem Paket

update
die Paketdatenbank aktualisieren

upgrade
ein einzelnes Paket oder das gesamte System aktualisieren

Anmerkung

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.


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


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.

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 6.2).

**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


**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 an⁴. Auf einem Ubuntu mit tasksel in der Version 2.88 sehen Sie diese Ausgabe:

**Ausgabe der Aufgabenbeschreibung eines tasks (Ubuntu)**

```
$ 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
Packages in 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)**

⁴Unter Debian 7 Wheezy ist die Ausgabe derzeit defekt und als Bug #756841 hinterlegt, siehe https://bugs.debian.org/756841
$ 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 8.34) mit aptitude download Paketname
- Pakete zur Installation, Aktualisierung oder Löschung vormerken (siehe Mit aptitude Vormerkungen machen unter Kapitel 11)
- Paket installieren (Abschnitt 8.38) mit aptitude install Paketname
- Paket in einer bestimmten Version halten (nicht aktualisieren) (siehe Kapitel 15 und Kapitel 16)
- Pakete deinstallieren (Abschnitt 8.43) mit aptitude remove Paketname
- Pakete erneut installieren (Abschnitt 8.39) mit aptitude reinstall Paketname
- installierte Paketliste oder die Veröffentlichung aktualisieren (Abschnitt 8.41) mit aptitude update, aptitude safe-upgrade und aptitude full-upgrade
- klären, warum ein Paket (nicht) installiert ist (Abschnitt 8.17) mit aptitude why Paketname bzw. aptitude why-not Paketname
- Paketabhängigkeiten mit aptitude search ?depends Paketname anzeigen (Abschnitt 8.19)

Dokumentation zu aptitude

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 Taste **Taste F10** aktivieren Sie bspw. die Menüleiste. Einige Terminals wie bspw. das Gnome-Terminal fangen die F10-Taste ab und belegen diese für die eigene Menüleiste. Über den Eintrag Bearbeiten → Menüfensterkombinationen 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 dar (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 10.12 ein.

<table>
<thead>
<tr>
<th>Farbkombination</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>schwarzer Hintergrund mit weißer Schrift</td>
<td>das Paket wird nicht verändert</td>
</tr>
<tr>
<td>roter Hintergrund mit weißer Schrift</td>
<td>das Paket ist defekt oder kann nicht installiert werden</td>
</tr>
<tr>
<td>blauer Hintergrund mit weißer Schrift</td>
<td>das Paket wird aktualisiert</td>
</tr>
<tr>
<td>weißer Hintergrund mit schwarzer Schrift</td>
<td>die Paketversion bleibt erhalten, kann jedoch aktualisiert werden</td>
</tr>
<tr>
<td>grüner Hintergrund mit schwarzer Schrift</td>
<td>Paket wird installiert</td>
</tr>
<tr>
<td>lila Hintergrund mit schwarzer Schrift</td>
<td>Paket wird entfernt („deinstalliert“)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Aktion</th>
<th>Tastenbelegung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilfe</td>
<td>?</td>
</tr>
<tr>
<td><strong>aptitude</strong> beenden (Vormerkungen werden gespeichert)</td>
<td>Shift+ q</td>
</tr>
<tr>
<td><strong>aptitude</strong> abbrechen (alle Vormerkungen gehen verloren)</td>
<td>Ctrl+kbd[ ]</td>
</tr>
<tr>
<td>Info-Fenster ein- und ausblenden</td>
<td>Shift+ D</td>
</tr>
<tr>
<td>Zwischen den Info-Ansichten wechseln</td>
<td>l</td>
</tr>
<tr>
<td>Zwischen beiden Fenstern hin- und herwechseln</td>
<td>Tab</td>
</tr>
<tr>
<td>In das Menü von <strong>aptitude</strong> wechseln</td>
<td>Ctrl+t oder F10</td>
</tr>
<tr>
<td>Paketlisten aktualisieren</td>
<td>u</td>
</tr>
<tr>
<td>Ausgewähltes Paket auswählen</td>
<td>-</td>
</tr>
<tr>
<td>Ausgewähltes Paket entfernen</td>
<td>-</td>
</tr>
</tbody>
</table>

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 zusätzli-
ches 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 \textit{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 \textit{aptitude}

Tabelle 6.4: Tasten zur interaktiven Konfliktlösung bei \textit{aptitude}

<table>
<thead>
<tr>
<th>Aktion</th>
<th>Tastenbelegung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vorschläge zur Konfliktlösung anzeigen</td>
<td>e</td>
</tr>
<tr>
<td>Nächsten Vorschlag anzeigen</td>
<td></td>
</tr>
<tr>
<td>Vorherigen Vorschlag anzeigen</td>
<td></td>
</tr>
<tr>
<td>Ersten Vorschlag anzeigen</td>
<td>&lt;</td>
</tr>
<tr>
<td>Letzten Vorschlag anzeigen</td>
<td>&gt;</td>
</tr>
<tr>
<td>Teilvorschlag akzeptieren</td>
<td>a</td>
</tr>
<tr>
<td>Teilvorschlag ablehnen („reject“)</td>
<td>r</td>
</tr>
<tr>
<td>Vorschlag anwenden</td>
<td>!</td>
</tr>
</tbody>
</table>

Darüber hinaus verfügt \textit{aptitude} über eine Ansicht, in der Sie Pakete nach \textit{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 \textit{Kategorienbrowser} gilt als veraltet\footnote{Es handelt sich dabei um eine hart in \textit{aptitude} verdrahtete und schon sehr lange nicht mehr gepflegte Kategorisierung der Pakete}, funktioniert seit einigen Ver-
sionen nicht mehr und wird voraussichtlich demnächst ganz entfernt [\textit{aptitude-categorical-browser-to-be-removed}]. Der oben
angerissene Debags-Browser ist der offizielle, wesentlich aktuellere und besser gepflegte Ersatz dafür.

\footnotetext[5]{Es handelt sich dabei um eine hart in \textit{aptitude} verdrahtete und schon sehr lange nicht mehr gepflegte Kategorisierung der Pakete}
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).

### 6.3.3 Nala

Das Werkzeug Nala [Debian-Paket-nala] ist bislang noch recht unbekannt und versteht sich als Frontend für APT. Ziel ist, eine übersichtlichere Darstellung vom aktuellen Paketbestand sowie bei dessen Änderungen zu erhalten, indem graphische Elemente in die Ausgabe einfließen. Abbildung 6.5 zeigt den Dialog auf der Kommandozeile zur Entfernern des Paketes `xsnow`.

![Abbildung 6.5: Entfernen des Paketes `xsnow` mittels Nala](image)

Das gesamte Verhalten und die Bedienung von Nala lehnt sich an DNF an - DNF für APT wäre somit eine gute Zusammenfassung. An Schaltern versteht es die Unterkommandos von APT, bspw. `install` zur Installation von Softwarepaketen, `remove` zum Löschen sowie `purge` und `remove --purge` zum vollständigen Löschen eines Softwarepakets. Mit dem Unterkommando `history` stöbern Sie in der Historie von Nala, sprich: Sie sehen daraus, welche Paketaktionen bereits vorher durchgeführt wurden.


### 6.4 GUI zur Paketverwaltung

#### 6.4.1 Synaptic

Abbildung 6.6: Softwareauswahl in `synaptic`

`Synaptic` bedienen Sie über die Menüleiste, eine Reihe von Knöpfen darunter und eine dreispaltige Paketübersicht. Die Darstellung konfigurieren Sie über die Menupunkte `Einstellungen` → `Einstellungen` und `Einstellungen` → `Werkzeugeiste`. Abbildung 6.7 zeigt als Beispiel das Dialogfenster, über welches Sie die Farben zum jeweiligen Installationsstatus eines Pakets festlegen.

Abbildung 6.7: 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 6.8 zeigt die Informationen zum Paket `ding`. 
Abbildung 6.8: 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 6.6 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. Über den rechten Knopf (Änderungsprotokoll abrufen) zeigt Ihnen *Synaptic* die Änderungsdatei (engl. *Changelog*) zum ausgewählten Paket an.


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 6.9, aus deren Liste 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.

---

6.4.2 Muon


Wählen Sie ein Paket mit einem Mausklick aus, so stellt Muon im Fensterbereich unter der Paketliste Metadaten und Details über das Paket dar. Dies umfaßt u.a. die Paketbeschreibung, die Paketabhängigkeiten, verfügbare Versionen, den Paketinhalt (Dateien im Paket), den Paketbetreuer, die installierte Größe der Software, die Downloadgröße und das Quellpaket, aus dem das Binärpaket gebaut wurde.

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 zu wajig (Abschnitt 8.44.6) und zu cupl (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 6.11 sehen Sie die zweispaltige Darstellung — links die Paketkategorien, rechts oben die Paketliste mit Paketname samt Versionsnummer und rechts unten die ausführliche Paketbeschreibung — hier 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 6.11: 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.


**Paketinformationen zu kexi in der Shell von SmartPM**

```
$ smart --shell
Smart Package Manager 1.4 - Shell Mode
```
Debian-Paketmanagement

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/
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.

This package is part of the Calligra Suite.

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 wurde³.


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 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 PolicyKit⁸ umzustellen. PackageKit ist nicht dafür gedacht, hochspezialisierte Paketverwaltungsoftware zu ersetzen.

³Bislang scheint SmartPM diese Markierungen nicht mit APT oder aptitude zu synchronisieren. Dieses Verhalten ist als Bug registriert.
⁸Berechtigungsdienst, der die Kommunikation von Software via DBus-Protokoll untereinander regelt.

Abbildung 6.12: Gnome Packagekit mit ausgewähltem Paket dctrl-tools


6.4.5 GDebi


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]: ...
```


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.


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 ersetzen10.

6.5 Webbasierte Programme

6.5.1 Ubuntu Landscape


9Verfügbar ab Debian 8 Jessie und Ubuntu 14.04 LTS Trusty Tahr
10Letzteres ist auch kein Wunder, da sowohl gdebi als auch diese Funktionalität von APT vom gleichen Autor stammen.
11Die Pakete dafür heißen landscape-client, landscape-client-ui, landscape-client-ui-install und landscape-common

Abbildung 6.15: 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 6.5.3) 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.

### 6.5.2 Appnr


### 6.5.3 Communtu


Die technische Basis hinter dem Angebot der Plattform bilden Metapakete (siehe Abschnitt 2.7.2) sowie das Werkzeug apturl (siehe Kapitel 25). Nachdem Sie über die Webseite Ihre gewünschte Software ausgewählt haben, wird automatisch ein erstes Metapaket erzeugt. Dieses beinhaltet lediglich die Abhängigkeiten zu den von Ihnen 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 auch einspielen kann.

6.5.4 Univention Corporate Server (UCS)


Den Ausgangspunkt bildet stets eine stabile Debian-Veröffentlichung — UCS 5.0 basiert z.B. auf Debian 10 Buster, UCS 4.4 auf Debian 9 Stretch. 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.


`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 6.18) 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 6.18: Univention App Centre](image-url)
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.


Abbildung 7.1: Heruntergeladene Pakete im Paketcache


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, *aptitude* fragt Sie hingegen als Benutzer, ob Sie trotzdem fortsetzen oder den Vorgang abbrechen möchten. Zu überlegen ist das beispielsweise, wenn nur ein einziges Paket fehlgeschlug, 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 insbesondere 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 Dateien im Paketcache

Der Paketcache hat keine komplexe Struktur. Darin befinden sich die folgenden Einträge:

Debianpakete
die Pakete in Form von deb-Dateien, die Sie zuvor mittels apt-get oder aptitude heruntergeladen haben

lock-Datei
ein Marker in Form einer leeren Datei. Dieser verhindert, dass zwei Programme — bspw. apt-get und Synaptic — nicht gleichzeitig einen Download eines Paketes versuchen.

das Verzeichnis partial
Debianpakete, die noch nicht vollständig heruntergeladen wurden. Wie Pakete darin landen, erklären wir Ihnen unter Abschnitt 7.1.1.

7.3 Paketcache-Status

Den aktuellen Zustand des Paketcaches erfahren Sie mit Hilfe des 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 Speicher: 73,0 k
Gesamtmenge an Speicher: 11,8 M
Total buckets in PkgHashTable: 196613
   Unused: 108138
   Used: 88475
   Utilization: 44.9996%
   Average entries: 1.33334
   Longest: 17
   Shortest: 1
Total buckets in GrpHashTable: 196613
   Unused: 101900
   Used: 94713
   Utilization: 48.1723%
   Average entries: 1.3668
   Longest: 7
   Shortest: 1
```

Die nachfolgenden Beschreibungen zu den einzelnen Zeilen der Ausgabe basieren auf der Manpage zu `apt-cache`. In Klamern finden sie die dazugehörige englische Übersetzung der Schlüsselworte. Findet sich als Beschreibung die Angabe `ToDo`, liegt noch keine Beschreibung vor — auch nicht in der Manpage zu `apt-cache`.

**Gesamtzahl an Paketnamen (total package names)**
gibt die Gesamtzahl der im Paketcache gefundenen Pakete an.

**Gesamtzahl an Paketstrukturen (total package structures)**

- **Gewöhnliche Pakete (normal packages)**

- **Rein virtuelle Pakete (pure virtual packages)**

- **Einzelne virtuelle Pakete (single virtual packagess)**
gibt die Anzahl der Pakete mit nur einem Paket an, das ein bestimmtes virtuelles Paket bereitstellt. Beispielsweise ist `X11-text-viewer` ein solches Paket, aber nur das Paket `xless` stellt dieses bereit.

- **Gemischte virtuelle Pakete (mixed virtual packages)**
gibt die Anzahl der Pakete an, die entweder ein bestimmtes virtuelles Paket bereitstellen oder den virtuellen Paketnamen als Paketnamen haben. Beispielsweise ist `debconf` solches Paket, dieses wird aber auch vom Paket `debconf-tiny` bereitgestellt.

- **Fehlende Pakete (missing)**
das benennt die Anzahl der Paketnamen, auf die eine Abhängigkeit verweist, die aber von keinem Paket bereitgestellt werden. Fehlende Pakete könnten auftauchen, wenn nicht auf eine vollständige Veröffentlichung zugegriffen oder ein (echtes oder virtuelles) Paket aus einer Veröffentlichung gestrichen wurde. Normalerweise erfolgt der Bezug in der Paketbeschreibung über die Schlüsselworte `Conflicts` oder `Breaks` (siehe dazu Abschnitt 4.1.1).

**Gesamtzahl an unterschiedlichen Versionen (total distinct versions)**
diese Angabe benennt die Anzahl der im Paketcache gefundenen Paketversionen. Dieser Wert ist meistens identisch mit der Anzahl der gesamten Paketnamen. Wenn auf mehr als eine Veröffentlichung zugegriffen wird — zum Beispiel `stable` und `unstable` zusammen —, kann dieser Wert deutlich größer als die gesamte Anzahl der Paketnamen sein.

**Gesamtzahl an unterschiedlichen Beschreibungen (total distinct descriptions)**
Angabe zur Anzahl unterschiedlicher Beschreibungen.

**Gesamtzahl an Abhängigkeiten (total dependencies)**
beschreibt die Anzahl der Abhängigkeitsbeziehungen, den alle Pakete im Paketcache beanspruchen.

**Gesamtzahl an Version/Datei-Beziehungen (total ver/file relations)**
`ToDo`

**Gesamtzahl an Beschreibung/Datei-Beziehungen (total desc/file relations)**
`ToDo`

**Gesamtzahl an Bereitstellungen (total provides mappings)**
`ToDo`

**Gesamtzahl an Mustern (total globbed strings)**
`ToDo`

**Gesamtmenge des Abhängigkeits-/Versionsspeichers**
`ToDo`

**Gesamtmenge an Slack (total slack space)**
`ToDo`
Gesamtmengen an Speicher (total space accounted for)
Angabe zur Gesamtgröße des Paketcaches.

Gesamtzahl der Einträge in der PkgHashTable (total buckets in PkgHashTable)
ToDo

Gesamtzahl der Einträge in der GrpHashTable (total buckets in GrpHashTable)
ToDo

Ist der Platz auf Ihrem Speichermedium knapp, sehen Sie in Zeile Gesamtmengen an Speicher die Angabe, welche Menge durch den Paketcache gerade belegt wird. Wie Sie darin wieder für Ordnung sorgen, lesen Sie unter „Paketcache aufräumen“ in Abschnitt 7.5 nach.

7.4 Größen des Paketcaches

7.4.1 Wieviel Platz belegt der Paketcache?

APT benutzt zwei Caches — einen im RAM, und einen auf einem (lokalen) Speichermedium. Die Informationen dazu bedurften aufwendiger Recherche.

Die Größe des APT-Caches im RAM erhalten Sie über den Aufruf von `apt-cache stats` in Kombination mit `grep`, wobei Sie die Zeile "Gesamtmengen an Speicher" aus der Ausgabe von `apt-cache stats` herausfiltern.

```
apt-cache stats liefert die Größe des APT-Caches im RAM

$ apt-cache stats | grep "Gesamtmengen an Speicher"
Gesamtmengen an Speicher: 31,1 M
$
```

Anmerkung
Verwenden Sie ein Debian GNU/Linux mit englischer Spracheinstellung, lautet die betreffende Zeile "Total space accounted for:.


```
du ermittelt die Größe des Verzeichnisses des Paketcaches

$ du -sh /var/cache/apt/archives/
1,4M /var/cache/apt/archives/
$
```

7.4.2 Größen des Paketcaches festlegen

Der Defaultwert ist mit 0 festgelegt, d.h. es besteht keine Größenbegrenzung seitens APT. In diesem Fall ergeben sich die Limits aus der physiskalischen Größe der Partition, der das Verzeichnis für den Paketcache zugeordnet ist sowie dem Dateisystem auf der Partition selbst. Im Dateisystem sind wiederum die Anzahl der Inodes (Einträge) und die Blockgröße festgelegt, die letztendlich regulen, wie viele Pakete als Datei gespeichert werden können.
7.5 Paketcache aufräumen

7.5.1 Weshalb aufräumen?


Den Paketcache finden Sie im Verzeichnis /var. Je nach Partitionierung Ihres Datenträgers ist der Bereich nicht separat und möglicherweise Bestandteil der /-Partition. Ist diese vollständig mit Daten gefüllt, funktioniert vieles auf Ihrem Linuxsystem nicht mehr.

Bei ausgefeilteren Installationen bestehen häufig separate Partitionen für /var oder /var/cache. Läuft 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.


7.5.2 Paketverwaltung passend konfigurieren

Die Programme dpkg, apt-get, aptitude und Synaptic räumen den Paketcache in der Standardeinstellung nicht eigenständig auf und belassen die Pakete nach der Installation oder Aktualisierung im Paketcache. Ob überhaupt, wann und insbesondere wie aufgeräumt wird, entscheiden Sie als Systembetreuer selbst und müssen dazu das von ihnen verwendete Werkzeug entsprechend konfigurieren.

Aus unserer Sicht sind die einzelnen Werkzeuge in Debian GNU/Linux wie folgt konfiguriert:

**bei dpkg**

Es belässt sowohl die .deb-Datei im Verzeichnis, als rührt auch den Paketcache nicht an.

**bei apt-get**

Die Pakete verbleiben im Paketcache und werden nicht daraus entfernt.

**bei aptitude**

Abbildung 7.2: Alte Pakete aus dem Paketcache mittels Aptitude entfernen

beiapt
Laden Sie ein oder mehrere Pakete mittels `apt install` oder `apt upgrade` herunter und installieren diese erfolgreich, entfernt `apt` diese Pakete danach aus dem Paketcache. Dieses Verhalten wird über den Eintrag `Clean-Installed` in `apt.conf` gesteuert, der Default-Wert dafür ist `on` und steht für das Aufräumen.

7.5.3 Kommandos zum Aufräumen


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**

```bash
# apt-get -o APT::Clean-Installed=off autoclean
...
```


**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**

```bash
# 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 7.3).

![Abbildung 7.3: Großreinemachen mit Hilfe von Synaptic](image)


### 7.5.4 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` (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.
7.5.5 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 clean

**Schalter --autoclean-on-startup**
entspricht dem Aufruf aptitude autoclean

Kapitel 8

Paketoperationen

8.1 Paketoperationen und deren Abfolge


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 8.34). 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 8.35 bis Abschnitt 8.46). Den Abschluss bildet der Vorgang, eine ganze Distribution auf den neuen Stand zu bringen oder zu einer anderen Veröffentlichung zu wechseln (siehe Abschnitt 8.47).

8.2 Paketlisten und Muster


Installation der Pakete xpdf, kile und cessed mittels APT in einem Aufruf

```
# apt-get install xpdf kile cessed
... 
#
```


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)
```

Debian-Paketmanagement

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"
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Architektur</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>libc6:i386</td>
<td>2.19-11</td>
<td>i386</td>
<td>GNU C Library: Shared libraries</td>
</tr>
</tbody>
</table>

### 8.3 Bekannte Paketnamen auflisten


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.

```
$ apt-cache pkgnames | sort | more
0ad
0ad-data
0ad-db
2ping
2vcard
3270-common
389-console
3dchess
...
$
```

Kennen Sie jedoch nur ein Fragment eines Paketnamens, gibt es verschiedene Wege. Der gebräuchliche Weg war für lange Zeit `apt-cache search -n Suchmuster`. Seit APT 1.0 kann man sich aber das `-cache` sparen und `apt search -n Suchmuster` verwenden. Allerdings sieht die Ausgabe jeweils unterschiedlich aus, u.a. sind bei `apt` die Paketnamen noch in grün hervorgehoben und die Ausgabe ist mehrzeilig pro Paket.

In beiden Fällen schränkt der Schalter `-n` die Suche auf Paketnamen inklusive bereitgestellter Paketnamen ein.
Will man **aptitude** verwenden, so braucht man statt dem `−n` an das Suchmuster selbst `−n` vorne dranhängen und quoten, damit es von der Kommandozeilen-Shell nicht als persönliches Verzeichnis eines Nutzers interpretiert wird: `aptitude search ‘−n ...’` (Das Leerzeichen nach `−n` ist optional. Im Beispiel unten wurde es weggelassen.) **Aptitude** zeigt dann allerdings bei Systemen mit mehr als einer installierten Architektur jedes Paket einmal pro Architektur an.

**Suche nach dem Muster lynx in Paketnamen.**

```
$ apt-cache search −n lynx
lynx − Klassischer Textmodus-Webbrowser (nicht graphisch)
lynx-common − Gemeinsame Dateien für Lynx
lynx-dbgsym − debug symbols for lynx
$ apt search −n lynx
Sortierung# Fertig
Volltextsuche# Fertig
lynx/bullseye,now 2.9.0dev.6-1 amd64 [Installiert, automatisch]
   Klassischer Textmodus-Webbrowser (nicht graphisch)
lynx-common/bullseye,now 2.9.0dev.6-1 all [Installiert, automatisch]
   Gemeinsame Dateien für Lynx
lynx-dbgsym/bullseye-debug,now 2.9.0dev.6-1 amd64 [installiert]
   debug symbols for lynx

$ aptitude search ‘−n lynx’
i A lynx − Klassischer Textmodus-Webbrowser (nicht graphisch)
p lynx:i386 − Klassischer Textmodus-Webbrowser (nicht graphisch)
i A lynx-common − Gemeinsame Dateien für Lynx
v lynx-common:i386 − Gemeinsame Dateien für Lynx
i lynx-dbgsym − debug symbols for lynx
p lynx-dbgsym:i386 − debug symbols for lynx
$
```

Weiß man den Anfang des Paketnamens, aber den Rest nicht, kann man auch noch **apt-cache pkgnames** mit Parameter verwenden oder aber eine der o.g. Suchen mit dem Anker `ˆ` zum Markieren des Anfangs der Zeichenkette:

**Suche nach dem Muster links am Anfang des Paketnamens.**

```
$ apt-cache pkgnames links
links2
links
links-dbgsym
links2-dbgsym
$ apt-cache search −n ‘^links’
links − Textmodus-Webbrowser
links2 − Webbrowser für den grafischen und den Textmodus
links-dbgsym − debug symbols for links
links2-dbgsym − debug symbols for links2
$ apt search −n ‘^links’
Sortierung# Fertig
Volltextsuche# Fertig
links/bullseye,now 2.21-1+b1 amd64 [Installiert, automatisch]
   Textmodus-Webbrowser
links-dbgsym/bullseye-debug,now 2.21-1+b1 amd64 [installiert]
   debug symbols for links
links2/bullseye,now 2.21-1+b1 amd64 [Installiert, automatisch]
   Webbrowser für den grafischen und den Textmodus
links2-dbgsym/bullseye,now 2.21-1+b1 amd64 [installiert]
   debug symbols for links2
$
```

```
$ aptitude search ‘−nˆlinks’
```

**Suche nach dem Muster `mlinks` am Ende eines Paketnamens**

```
$ apt-cache pkgnames | egrep 'mlinks$'
symlinks
python3-sphinx-paramlinks

$ apt-cache search -n 'mlinks$'
python3-sphinx-paramlinks - Sphinx extension to make param links linkable (Python 3 version)

$ apt search -n 'mlinks$'
symlinks/unstable,testing,now 1.4-4 amd64 [Installiert, automatisch]
  Scannen und Ändern symbolischer Links
$ aptitude search '~n mlinks$'
p python3-sphinx-paramlinks - Sphinx extension to make param links linkable (Python
i A symlinks - Scannen und Ändern symbolischer Links
p symlinks:i386 - Scannen und Ändern symbolischer Links
```

Daß eine Suche nach "links" nicht nur am Anfang oder Ende des Paketnamens zu wesentlich mehr Treffern führt, sei dem Leser als Übung überlassen.

Will man zwingend nur die Paketnamen sehen, gibt es ebenfalls mehrere Möglichkeiten:

- O.g. Ausgaben von `apt-cache mittels awk,cut oder sed nach dem ersten Leerzeichen abschneiden.`

- Die Ausgabe von `apt-cache pkgnames mittels grep o.ä. durchsuchen.`


Da `dglob` einiges anders macht als in den o.g. Paketen, schauen wir es uns hier getrennt an.

Das Werkzeug `dglob` setzt auf `grep-aptavail` und `grep-dctrl` auf um die von APT heruntergeladenen Paketlisten zu durchsuchen. Im Gegensatz zu `apt, apt-cache und aptitude` arbeitet es nicht mit regulären Ausdrücken sondern mit Platzhalten (engl. "wildcards"). Dies ist ähnlich zu `dpkg -l`, allerdings sind Platzhalter für beliebige Zeichenketten an Anfang und Ende des Suchmusters bei `dglob` implizit.

Ebenfalls ähnlich zu `dpkg -l listet dglob ohne weitere Schalter nur installierte Pakete auf. Mit dem Schalter `--a` ändern Sie seine Verhaltensweise dahingehend, daß es alle bekannten Paketen in Erwägung zieht — unabhängig davon, ob diese jeweils auf Ihrem System installiert sind oder nicht. Ohne die Option beschränkt sich `dglob` nur auf die bereits installierten Pakete.

**Suche nach elinks mit dglob, dglob -a und dpkg -l im Vergleich**
$ dpkg -l elinks
   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 elinks 0.13.2-1+b1 amd64 advanced text-mode WWW browser

$ dpkg -l '*elinks*'
   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 elinks 0.13.2-1+b1 amd64 advanced text-mode WWW browser
   ii elinks-data 0.13.2-1 all advanced text-mode WWW browser - data files
   un elinks-doc <keine> <keine> (keine Beschreibung vorhanden)
   un elinks-lite <keine> <keine> (keine Beschreibung vorhanden)

$ dglob elinks
   elinks:amd64
   elinks-data:all
   $ dglob -a elinks
   elinks:amd64
   elinks-data:all
   elinks-doc:all
   elinks:i386
   elinks-dbgsym:amd64
   elinks-dbgsym:i386

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.


8.4.1 dpkg -s Paketname und dlocate -s Paketname


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 8.32.1).

**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: 25eb706b210a165f9ae3a149338c129c383b82df
SHA256: b41970322366d8a8fd174aa32b223dd54d05e4ab1dafddd97390e0fc5f17ed41
```

8.4.4 apt-cache showpkg Paketname


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 letztnannte 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:
    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: de
File: /var/lib/apt/lists/ftp.de.debian.org_debian_dists_jessie_main_binary- amd64_Packages
MD5: 2ebe91e411d46309a61861db507e5c2f

Description Language: en
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+.deb8u1 (/var/lib/dpkg/status)

Description Language: de
File: /var/lib/apt/lists/ftp.de.debian.org_debian_dists_jessie_main_binary- amd64_Packages
MD5: 2ebe91e411d46309a61861db507e5c2f

Description Language: en
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) liblzma2-2 (0 (null))
libpam0g (2 0.99.7.1) libpamkcs11-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)) openssl1 (0 (null))
pants: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) liblzma2-2 (0 (null)) ←
      libpam0g (2 0.99.7.1) libpamcall-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 `xdpdf`.

### Pakete, deren Zustand gehalten wird

```
# apt-mark hold xpdf
xdpdf auf Halten gesetzt.
# apt-mark showhold xpdf
xdpdf
# apt-mark unhold xpdf
Halten-Markierung für xpdf entfernt.
```


## 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 mittlerweile drei grundlegende Möglichkeiten. Einerseits ist das `dpkg -l`, andererseits `aptitude search '~i'` und drittens `apt list --installed`. 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. Intern leitet `dpkg` den Aufruf an das Abfragewerkzeug `dpkg-query` weiter.

#### 8.5.1.1 Ausgabe von `dpkg` interpretieren

Tabelle 8.1: Paketzustand deuten

<table>
<thead>
<tr>
<th>gewünschte Aktion durch den Paketmanager</th>
<th>Paketzustand</th>
<th>Fehler-Schalter</th>
</tr>
</thead>
<tbody>
<tr>
<td>u: Unbekannt (<em>unknown</em>)</td>
<td>n: Paket ist nicht installiert (<em>not installed</em>)</td>
<td>(leer): kein Fehler</td>
</tr>
<tr>
<td>i: Installieren (<em>installed</em>)</td>
<td>c: nur die Konfigurationsdatei ist vorhanden (<em>configured</em>)</td>
<td>R: eine Neuinstallation ist notwendig (<em>reinstall</em>)</td>
</tr>
<tr>
<td>h: Halten (<em>hold</em>)</td>
<td>H: Paket ist nur halb installiert (<em>half installed</em>)</td>
<td></td>
</tr>
<tr>
<td>r: Entfernen (<em>remove</em>)</td>
<td>U: Paket wurde entpackt (<em>unpacked</em>)</td>
<td></td>
</tr>
<tr>
<td>p: Vollständig Löschen (<em>purge</em>)</td>
<td>F: Fehlgeschlagene Konfiguration (<em>failed</em>)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>W: Trigger erwartet (<em>aWaited trigger</em>)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>t: Trigger anhängig (<em>trigger</em>)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>i: Installiert (<em>installed</em>)</td>
<td></td>
</tr>
</tbody>
</table>


Tabelle 8.2: Alltägliche Kombinationen zum Paketzustand

<table>
<thead>
<tr>
<th>Paketzustand</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>ii</td>
<td>das Paket ist vollständig und fehlerfrei installiert sowie konfiguriert.</td>
</tr>
<tr>
<td>rc</td>
<td>das Paket wurde gelöscht, aber die Konfigurationsdateien sind noch gespeichert (Abkürzung für removed, configured, no error).</td>
</tr>
<tr>
<td>un</td>
<td>unbekanntes, nicht installiertes Paket (Abkürzung für unknown, not installed).</td>
</tr>
<tr>
<td>hi</td>
<td>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).</td>
</tr>
</tbody>
</table>

8.5.1.2 Vollständige Paketliste anzeigen

Ohne die Angabe des Paketnamens zeigen Sie alle Pakete und Paketreste an, die auf Ihrem System derzeit installiert sind. In nachfolgender Paketliste zeigen die einzelnen Spalten nacheinander den Paketzustand, den Namen des Pakets (siehe Abschnitt 2.11), dessen Version und die dazugehörige 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)
<table>
<thead>
<tr>
<th>// Name</th>
<th>Version</th>
<th>Beschreibung</th>
</tr>
</thead>
</table>
+++-=========================================-====================================-==========================
ii a2ps 1:4.14-1.1 GNU a2ps ←
| 'Anything to Po |
```

8.5.1.3 Liste anhand von Filtern einschränken

dpkg akzeptiert Paketnamen als Filter. Geben Sie ein oder mehrere Pakete als Parameter durch Leerzeichen getrennt an, erhalten Sie nur Informationen zu diesen benannten Paketen. Das nachfolgende Beispiel listet ausschließlich die Informationen zum Paket *map auf.

Informationen zu einem ausgewählten Paket anzeigen

$ dpkg -l nmap

Suche nach Paketen anhand eines Musters

$ dpkg -l '*map'

Eine Einschränkung auf bestimmte Zeichen erreichen Sie über eckige Klammern im Muster. Die Angabe *\{nt\}map passt auf alle Namen bestehend aus beliebigen Zeichen (\{nt\}) gefolgt von einer * oder *t ([nt]) und der Zeichenkette map. Gefunden werden die drei Pakete bitmap, nmap und zenmap, wobei nur das Paket nmap installiert ist und die anderen beiden Pakete nicht.

Suche nach Paketen anhand eines ausgefeilten Musters

$ dpkg -l '*[nt]*map'

Eine Einschränkung auf bestimmte Zeichen erreichen Sie über eckige Klammern im Muster. Die Angabe *\{nt\}map passt auf alle Namen bestehend aus beliebigen Zeichen (*\{nt\}) gefolgt von einem n oder t ([nt]) und der Zeichenkette map. Gefunden werden die drei Pakete bitmap, nmap und zenmap, wobei nur das Paket nmap installiert ist und die anderen beiden Pakete nicht.
8.5.1.4 Ausgabe minimieren

Geht es Ihnen lediglich um die Namen der derzeit vollständig installierten Pakete ohne deren Versionsnummer und Beschreibung, haben Sie drei Möglichkeiten zur Lösung — a) dpkg-query mit einem speziellen Formatstring für die Ausgabe in Kombination mit egrep, awk und sort, b) die Kombination aus dpkg, egrep, awk und sort sowie c) die Verwendung von dpkg-awk. dpkg selbst bietet von sich aus bislang keinen entsprechenden, einzelnen Schalter an, der diese spezifische Ausgabe ermöglicht.

Für den Fall a) nutzen Sie den Schalter --show und den Formatstring --format="\n". Dabei kürzt -f die Langform --showformat ab. Die Angabe '${db:Status-Abbrev}' liefert Ihnen den Installationsstatus des Pakets und ${binary:Package}\n' den Paketnamen aus der Paketbeschreibung samt Zeilenumbruch am Ende der Zeile. Der vollständige Aufruf ist dann wie folgt:

Nur die Paketnamen ausgeben (dpkg-query)

$ dpkg-query -W -f='${db:Status-Abbrev} ${binary:Package}\n' * | egrep '^ii' | awk '{ print $2 }' | sort

Nur die Paketnamen ausgeben (dpkg, egrep, awk und sort kombiniert)

$ dpkg -l | egrep '^ii' | awk '{print $2}' | sort

Der dritte Fall c) benutzt das Werkzeug dpkg-awk aus dem gleichnamigen Paket [Debian-Paket-dpkg-awk]. Das Paket gehört nicht zur Standardinstallation und ist daher von Ihnen vor der Benutzung nachzustallieren. dpkg-awk wertet die beiden Dateien /var/lib/dpkg/status und /var/lib/dpkg/available aus.

Mit dem nachfolgenden Aufruf erhalten Sie eine Liste aller Pakete, die auf ihrem System gerade installiert sind. dpkg-awk filtert alle Zeilen heraus, auf die der reguläre Ausdruck Status: .* installed$ passt. Mit der Angabe Package weisen Sie dpkg-awk an, nur in die Felder mit den Paketnamen zu durchsuchen.

Die Liste der installierten Pakete mit dpkg-awk ermitteln

$ dpkg-awk "Status: .* installed$ -- Package
Package: libasan0
Package: libvorbisfile3
Package: libquadmath0
Package: libxkbfile1
...
Obige Ausgabe bearbeiten Sie mit UNIX/Linux-Tools weiter, bspw. mit cut oder sort, um die Ausgabe ihren Wünschen anzupassen.

### 8.5.2 Suche mit aptitude


Ergebnis ist eine Liste aller installierten Pakete. Daraus filtern wir im nachfolgenden Beispiel mittels grep nur die Zeilen heraus, in denen das Wort texlive enthalten ist.

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

- texlive - TeX Live: Eine anständige Auswahl der TeX-
- texlive-base - TeX Live: Grundlegende Programme und Datei
- texlive-bibtex-extra - TeX Live: Extra BibTeX styles
- texlive-binaries - Binärdateien für TeX Live
- texlive-common - TeX Live: Basiskomponenten
- texlive-doc-base - TeX Live: Dokumentation für TeX Live
```

Geht es Ihnen nur um die Namen der installierten Pakete auf ihrem System, hilft folgende Kombination aus aptitude, sed und awk weiter:

#### Paketliste mittels aptitude ausgeben

```
$ aptitude search '~i' | sed -E 's/i \[A \]? //' | awk '{ print $1 }'

aapt
acl
acpi
adduser
adwaita-icon-theme
alsa-base
...
```

Die Angabe -E ‘s/i [A ]? //’ bei sed aktiviert zunächst erweiterte Reguläre Ausdrücke (Schalter -E) und ersetzt danach alle Vorkommen der Zeichenkette aus einem kleinen i gefolgt von einem Leerzeichen, einem möglichen A oder Leerzeichen sowie einem abschließenden Leerzeichen durch nichts (es löscht die Zeichen aus der Zeile). Die Angabe von ‘{ print $1 } ’ bei awk gibt danach lediglich erste Spalte jeder Zeile aus, die den Paketnamen enthält. Alle weiteren Informationen, die aptitude bereitgestellt hatte, entfallen.

### 8.5.3 apt list --installed


#### 8.5.3.1 Vollständige Paketliste anzeigen

Ohne weitere Angaben erhalten Sie eine vollständige Paketliste. Die Ausgabe sieht dann so aus:

```
apt listet die installierten Pakete auf
```

$ apt list --installed

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Auflistung...
aapt/oldstable,now 1:7.0.0+r33-1 amd64 [Installiert,automatisch]
acpi/oldstable,now 2.2.52-3+b1 amd64 [installiert]
adduser/oldstable,now 3.115 all [installiert]
adwaita-icon-theme/oldstable,now 3.22.0-1+deb9u1 all [Installiert,automatisch]
...
$

8.5.3.2 Liste anhand von Filtern einschränken

Analog zu dpkg kann auch apt mit Filtern umgehen, um darüber die Ausgabe einzuschränken. apt akzeptiert dazu als Parameter sowohl eine Liste von Paketnamen, als auch Muster. Das nachfolgende Beispiel schränkt die Ausgabe auf die beiden Pakete nmap und nmap-common ein.

Liste auf bestimmte Pakete eingrenzen

$ apt list --installed nmap nmap-common

Auflistung... Fertig
nmap-common/stable,stable,now 7.93+dfsg1-1 all [Installiert,automatisch]
nmap/stable,now 7.93+dfsg1-1 amd64 [installiert]
$


Liste mit der Angabe eines Musters eingrenzen

$ apt list --installed '*map*'

Auflistung... Fertig
libchromaprint1/stable,now 1.5.1-2+b1 amd64 [Installiert,automatisch]
lidudevmapper/event1.02.1/stable,now 2:1.02.185-2 amd64 [Installiert,automatisch]
lidudevmapper/event1.02.1/stable,now 2:1.02.185-2 amd64 [installiert]
libunicode-map-perl/stable,now 0.112-13+b1 amd64 [Installiert,automatisch]
nmap-common/stable,stable,now 7.93+dfsg1-1 all [Installiert,automatisch]
nmap/stable,now 7.93+dfsg1-1 amd64 [installiert]
xbitmaps/stable,stable,now 1.1.1-2.2 all [Installiert,automatisch]
$

8.5.3.3 Ausgabe minimieren

Um die Ausgabe auf die Paketnamen einzугrenzen, helfen Ihnen wiederum die beiden Werkzeuge awk und tail weiter. awk filtert den Paketnamen aus jeder Zeile heraus und tail entsorgt die ersten vier Zeilen inklusive der Warnung. Die Angabe 2>1 hinter apt lenkt zuvor noch den Fehlerkanal stderr auf die Standardausgabe stdout um.

apt listet die installierten Pakete auf (Paketliste)

$ apt list --installed 2>1 | awk -F '/ / ' '{ print $1 } ' | tail +4
apt
acl
acpi
adduser
adwaita-icon-theme
...
$
8.5.4  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 Paketstatus, 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 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.


**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-pae 3.2+46 i386 Header files for Linux 686-pae ←
configuration (meta-package)
ii linux-image-2.6-686 3.2+46 i386 Linux for modern PCs (dummy package)
```
8.7 Liste der installierten, nicht-freien Pakete anzeigen

8.7.1 Recherche mit aptitude

Mit Hilfe einer spezialisierten Suchanfrage ermöglicht *aptitude* das Aufspüren nicht-freier Pakete. Es erwartet die Angabe eines Bereichs mit Hilfe des Schlüsselwortes *section* gefolgt vom Namen des Bereichs. Hier listen wir beispielhaft alle installierten Pakete aus dem Bereich *non-free-firmware* auf.

*Suche nach installierten Paketen aus dem Bereich non-free-firmware*

```
$ aptitude search '~i ?section(non-free-firmware)'
i amd64-microcode - Processor microcode firmware for AMD CPUs
i firmware-amd-graphics - Binary firmware for AMD/ATI graphics chips
i firmware-iwlwifi - Binary firmware for Intel Wireless cards
i firmware-realtek - Binary firmware for Realtek wired/wifi/BT adapters
```

Zudem können mehrere Bereiche in einer Anfrage miteinander kombiniert werden. Hier sind es die zwei Bereiche *non-free* und *non-free-firmware*.

*Suche nach installierten Paketen aus den beiden Bereichen non-free und non-free-firmware*

```
$ aptitude search '~i ?section(non-free) |~i ?section(non-free-firmware)'
i amd64-microcode - Processor microcode firmware for AMD CPUs
i firmware-amd-graphics - Binary firmware for AMD/ATI graphics chips
i firmware-iwlwifi - Binary firmware for Intel Wireless cards
i firmware-realtek - Binary firmware for Realtek wired/wifi/BT adapters
i icc-profiles - ICC color profiles for use with color profile aware software
```

*aptitude* unterstützt auch Wildcards. Damit vereinfacht sich obiger Aufruf wie folgt:

*Vereinfachter Aufruf zur Suche mit Wildcards*

```
$ aptitude search '~i ?section(non-free*)'
...
```

8.7.2 Recherche mit check-dfsg-status (vormals vrms)


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
```
Debian-Paketmanagement

<table>
<thead>
<tr>
<th>firmware-iwlwifi</th>
<th>Binary firmware for Intel PRO/Wireless 3945 and 802.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>nautilus-dropbox</td>
<td>Dropbox integration for Nautilus</td>
</tr>
<tr>
<td>openttd-opensfx</td>
<td>sound set for use with the OpenTTD game</td>
</tr>
<tr>
<td>opera</td>
<td>Fast and secure web browser and Internet suite</td>
</tr>
<tr>
<td>skype</td>
<td>Skype</td>
</tr>
<tr>
<td>unrar</td>
<td>Unarchiver for .rar files (non-free version)</td>
</tr>
</tbody>
</table>

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.

Ausgabe von check-dfsg-status auf einem Desktop-System (Debian 12 Bookworm)

```
$ check-dfsg-status
Non-free packages installed on c6
cpp-12-doc documentation for the GNU C preprocessor (cpp)
gcc-12-doc documentation for the GNU compilers (gcc, gobjc, g++)
gdb-doc The GNU Debugger Documentation
intel-media-va-driver-non-free VAAP driver for the Intel GEN8+ Graphics family
intel-microcode Processor microcode firmware for Intel CPUs
libretro-snes9x Libretro wrapper for Snes9x

Reason: No commercial distribution
manpages-posix Manual pages about using POSIX system
manpages-posix-dev Manual pages about using a POSIX system for developments
tar-doc documentation for the tar package
tegrarcm Tool to upload payloads in Tegra SoC recovery mode
unrar Unarchiver for .rar files (non-free version)
wap-wml-tools Wireless Markup Language development and test tools

Contrib packages installed on c6
anbox Android in a box
cpp-doc documentation for the GNU C preprocessor (cpp)
gcc-doc documentation for the GNU compilers (gcc, gobjc, g++)
gcc-doc-base several GNU manual pages
libdvd-pkg DVD-Video playing library - installer
torbrowser-launcher helps download and run the Tor Browser Bundle

12 non-free packages, 0.2% of 6022 installed packages.
6 contrib packages, 0.1% of 6022 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
idraw-parts
skype
```
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.


**Auflistung aller neuen Pakete mittels aptitude (Auszug)**

```sh
$ 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
... $```

Die Werkzeuge `dpkg`, `apt` und `apt-get` verfügen nicht über einen entsprechenden Schalter, um neue Pakete aufzulisten.

8.9 Pakete nach Prioritäten finden

Wie bereits in „Paket-Priorität und essentielle Pakete“ in Abschnitt 2.13 besprochen, ist jedem Debianpaket eine bestimmte Wichtigkeit zugeordnet. Dazu zählen erforderlich (`required`), wichtig (`important`), standard (`standard`), optional (`optional`), extra (`extra`) und die Markierung essentiell (`essential`).


**Auflistung der extra-Pakete durch aptitude**

```sh
$ aptitude search '?priority(extra)'
```

```
p 0ad-dbg - Echtzeit-Strategiespiel über antike Kriegs
p 389-console - 389 Management Console
p 389-console - 389 Management Console
p 4digits - Zahlenratespiel oder „Bulls and Cow“
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**

```sh
$ aptitude search '?essential'
```

```
p 389-console - 389 Management Console
p 4digits - Zahlenratespiel oder „Bulls and Cow“
p 4store - Engine zur Datenbankspeicherung und -abfra
p 7kaa-dbg - Seven Kingdoms Ancient Adversaries - Debug
... $```

Die Werkzeuge `dpkg`, `apt` und `apt-get` verfügen nicht über einen entsprechenden Schalter, um neue Pakete aufzulisten.
$ 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 SHeIl
  i bsdutils - Minimale Auswahl von Kommandos aus 4.4BSD-1I
  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 utility-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 anseht und sich deswegen selbst nicht deinstallieren will. Da Aptitude zum Teil APTs Bibliotheken benutzt, sieht es diesen Fall genauso¹.

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


Automatisch installierte Pakete mit `apt-mark` anzeigen

$ apt-mark showauto '.*tex$'
jadetex
luatex
texlive-luatex
$

¹Bis 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 Abhängigkeitskonflikt angeboten.
Eine ausführliche Beschreibung zu den genutzten Paketflags sowie ein Beispiel für das Aufspüren manuell installierter Pakete mit `apt-mark` erhalten Sie in Abschnitt 2.15.4.

### 8.10.2 aptitude benutzen


**Automatisch installierte Pakete mit aptitude anzeigen**

```
$ aptitude search '~M' | more
```

Die gleiche Liste erhalten Sie über die Textoberfläche. Dazu filtern Sie die Paketliste mit der Taste I 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](image)

### 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 Linux-Kern, 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.


8.11.1 Recherche auf der Kommandozeile


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

```bash
$ aptitude search '~o'
i cupswrapperhl2250dn - Brother HL2250DN CUPS wrapper driver
i foxtreader - FoxitReader is a browsing program designed for read
i gtkdiskfree - A program to show free/used space on filesystems
i hl2250dnpr - 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.


8.11.3 Umgang mit diesen Paketen


8.12 Aktualisierbare Pakete anzeigen

Sowohl APT als auch aptitude zeigen Ihnen an, für welche Pakete eine neuere Version bereitsteht. Alle Werkzeuge bieten dafür recht unterschiedliche Parameter und Ausgaben auf dem Terminal.

8.12.1 apt-get verwenden

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 mit apt-get, 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 libc6-1686 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-image-3.2.0-4-common
virtualbox-guest-source virtualbox-guest-utils virtualbox-guest-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 ]
...
#

8.12.2 apt benutzen

Das Werkzeug apt kennt für diesen Fall das Unterkommando list und den Schalter --upgradable. In der Praxis sieht das wie folgt aus (die nachfolgende Ausgabe stammt von einem Ubuntu 18 Bionic):

Aktualisierbare Pakete mit apt anzeigen

```bash
$ apt list --upgradable
Auflistung... Fertig
aspell/bionic-updates,bionic-security 0.60.7~20110707-4ubuntu0.1 amd64 [aktualisierbar von ← : 0.60.7~20110707-4]
distro-info-data/bionic-updates,bionic-updates,bionic-security,bionic-security 0.37ubuntu0 ← .6 all [aktualisierbar von~: 0.37ubuntu0.5]
...
```

8.12.3 aptitude verwenden

aptitude kennt für diesen Zweck die Suchoption ~U. Diese steht als Kurzform für ?upgradable.

Aktualisierbare Pakete mit aptitude anzeigen

```bash
$ 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
...
```

8.12.4 Synaptic verwenden

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).
8.13 Verfügbare Versionen eines Paketes anzeigen

8.13.1 aptitude verwenden

aptitude bietet Ihnen mehrere Wege an, um diese Frage zu beantworten. Zur Verfügung stehen die beiden Unterkommandos `versions` und `search`.


### 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
```


### Kompakte Ausgabe ohne Paketname

```
$ aptitude versions 'tzdata-java' --show-package-names=never --group-by=none
```

**Ausgabe von aptitude zur Liste der aktualisierbaren Pakete**

```bash
$ aptitude search '~U'
i acpi-support - Skripte zur Verwaltung von ACPI-Ereignissen
i acpi-support-base - Skripte zur Verarbeitung grundlegender ACPI-Ereignisse
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: Entwicklungsbinärdaten
i libc6 - Die »Embedded GNU C Library«: Laufzeitbibliotheken
i A libc6-dev - Die »Embedded GNU C Library«: Entwicklungsbibliotheken
i libc6-i686 - »Embedded GNU C Library«: Laufzeitbibliotheken [optimi
i A libmozjs24d - Mozilla SpiderMonkey JavaScript library
i A xulrunner-24.0 - XUL + XPCOM application runner
$```

**8.13.2 Mit apt stöbern**


**Status der installierten Pakete anzeigen**

```bash
$ apt list --all-versions mc xpdf
Aufstellung... Fertig
mc/oldstable,now 3:4.8.18-1 amd64 [Konfiguration-verbleibend]
xpdf/oldstable,now 3.04-4 amd64 [installiert]
$```

**8.13.3 apt-show-versions verwenden**


**Kompakte Ausgabe mittels apt-show-versions**

```bash
$ apt-show-versions base-files
base-files:amd64/jessie 8+deb8u2 upgradeable to 8+deb8u3
$```
Gibt es keine aktuellere Version, sehen Sie die folgende Ausgabe:

**Ausgabe von apt-show-versions für ein aktuelles Paket**

```
$ apt-show-versions xpdf
xpdf:amd64/stretch 3.04-4 updoate
$
```

Ist das Paket jedoch noch nicht auf ihrem System installiert, ist es etwas schmallippig:

**Ausgabe von apt-show-versions für ein noch nicht installiertes Paket**

```
$ apt-show-versions mc
mc:amd64 not installed
$
```

### 8.13.4 apt-cache benutzen


**Ausgabe von apt-cache mit dem Unterkommando madison für apt-doc** unter Debian 9 *Stretch* auf einem System mit der Architektur *i386*

```
$ apt-cache madison apt-doc
  apt-doc | 1.4.6 | http://ftp.ch.debian.org/debian stretch/main i386 Packages
  apt   | 1.4.6 | http://ftp.ch.debian.org/debian stretch/main Sources
$
```

### 8.13.5 rmadison einsetzen

Möchten Sie hingegen wissen, welche Versionen für die jeweiligen Veröffentlichungen bereitstehen, hilft Ihnen das Werkzeug *rmadison* aus dem Paket *devscripts* weiter [Debian-Paket-devscripts]. 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.

**Auflistung der verfügbaren Paketversionen mit rmadison**

```
$ rmadison apt-doc
  apt-doc | 0.9.7.9+deb7u7 | oldoldstable | all
  apt-doc | 1.0.9.8  | oldstable-kfreebsd | all
  apt-doc | 1.0.9.8.4 | oldstable | all
  apt-doc | 1.4.6   | stable | all
  apt-doc | 1.4.6   | testing | all
  apt-doc | 1.5~beta1 | unstable | all
$
```

Obige Ausgabe umfaßt die vier Spalten Paketname, Versionsnummer, Veröffentlichung und Architektur. Möchten Sie die Ausgabe hingegen auf eine bestimmte Veröffentlichung oder Architektur einschränken, akzeptiert *rmadison* die Schalter *-a* Architektur und *-s* Veröffentlichung. Um zu sehen, welche Version des Paketes *base-files* für die Veröffentlichung Debian 9 *Stretch* und die Architektur *amd64* bereitstehen, nutzen Sie den folgenden Aufruf:

**Gefilterte Auflistung der verfügbaren Paketversionen mit rmadison**

```
$ rmadison -s stretch -a amd64 base-files
  base-files | 9.9   | stable | amd64
$
```
8.13.6 grep-available und grep-aptavail benutzen


Paketinformationen zu xpdf

$ grep-aptavail -PX xpdf
Package: xpdf
Version: 3.04-4
Installed-Size: 371
Maintainer: Debian QA Group <packages@qa.debian.org>
Architecture: amd64
Replaces: xpdf-common, xpdf-reader
Provides: pdf-viewer
Depends: libc6 (>= 2.4), libgcc1 (>= 1:3.0), libpoppler64 (>= 0.48.0), libstdc++6 (>= 5), ←
   libx11-6, libxm4 (>= 2.3.4), libxt6
Recommends: poppler-utils, poppler-data, gsfonts-x11, cups-bsd
Conflicts: xpdf-common, xpdf-reader
Description: Portable Document Format (PDF) reader
Homepage: http://www.foolabs.com/xpdf
Description-md5: fa7a14f325304cc49bbc0086a88d330e
Tag: implemented-in::c++, interface::graphical, interface::x11,
   role::program, scope::application, uitoollkit::motif, use::viewing,
   works-with-format::pdf, works-with::text, x11::application
Section: text
Priority: optional
Filename: pool/main/x/xpdf/xpdf_3.04-4_amd64.deb
Size: 159144
MD5sum: 8341b3ced6214b35185fdb42d8e7dcd7
SHA256: 926673359583d0e4ed1f57774642303e5fed5d95ad90b5debde6df4c43e8237
$

8.14 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.

8.14.1 Paketquellen untersuchen mit apt-cache policy


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
8.14.2 Informationen für ein bestimmtes Paket erhalten

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**

```bash
$ apt-cache policy gdm3

gdm3:
  Installiert: 3.4.1-8
  Installationskandidat: 3.4.1-8
  Versionstabelle:
** 3.4.1-8
  500 http://ftp.de.debian.org/debian/ wheezy/main i386 Packages

$ 
```

**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**

```bash
$ apt-cache policy linux-libc-dev

linux-libc-dev:
  Installiert: 3.2.51-1
  Installationskandidat: 3.2.57-3
  Versionstabelle:
    3.2.57-3
    500 http://ftp.de.debian.org/debian/ wheezy/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

```bash
$ 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
```

### 8.14.3 Verfügbare Paketversionen ermitteln

Hier spielen die Programme `apt-cache`, `rmadison`, `aptitude`, `apt` und `apt-show-versions` ihre Stärken aus. Darauf gehen wir ausführlich in Abschnitt 8.13 ein.

### 8.15 Installationsgröße eines Pakets


#### Ermittlung der Installationsgröße des Pakets `htop` mit `dlocate`

```bash
$ 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
```


#### Ausgabe der Installationsgröße eines Pakets mit `aptitude`

```bash
# aptitude -Z install aptitude-doc-it aptitude-doc-ru
Die folgenden NEUEN Pakete werden zusätzlich installiert:
```
8.16 Größtes installiertes Paket finden

8.16.1 dpkg


Aufspürung der zehn größten installierten Softwarepakete

```bash
$ dpkg-query -Wf '{Installed-size}	{Package}\n' | column -t | sort -nr | head
1414534 texlive-fonts-extra
398647 linux-image-6.1.0-12-amd64
398283 linux-image-6.1.0-10-amd64
334238 fonts-noto-extra
271679 llvm-14-dev
217530 firefox-esr
206065 texlive-lang-english
187651 openjdk-17-jre-headless
168399 pandoc
138224 libboost1.74-dev
```

8.16.2 dpigs


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

```bash
$ 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
```

³ „pig“ ist Englisch für Schwein bzw. Sau. Es geht sozusagen um Debianpakete, die den Plattenplatz versauen, auch bekannt als „Plattenplatzschweine“.
dpigs verfügt nur über einige wenige, aber durchaus nützliche Schalter. Um beispielsweise die Anzahl der ausgegebenen Pakete zu begrenzen, nutzen Sie die Option \texttt{-nZahl} (Langform \texttt{--lines=Zahl}), wobei \texttt{Zahl} der Wert für die gewünschte Anzahl ist. Der Schalter \texttt{-S} zeigt die Pakete an, die hingegen die größten Quelldaten haben. Mit dem Schalter \texttt{-H} rechnet dpigs die Größenangaben in menschlich lesbare Werte um. Der Schalter \texttt{-H} steht hierbei als Abkürzung für \textit{human readable}. Kombinieren Sie die drei letztgenannten Schalter, sieht die Recherche nach den fünf Paketen, die am meisten Quellcode benötigen, folgendermaßen aus:

\textbf{Ausgabe von dpigs mit Einschränkung auf die fünf Spitzenreiter für den Quelltext}

\begin{verbatim}
$ dpigs -S -n5 -H
  1.0G texlive-extra
  299.7M libreoffice
  274.7M texlive-base
  131.4M chromium-browser
  120.5M calligra
$
\end{verbatim}

8.16.3 \texttt{aptitude}

\texttt{aptitude} kann mit dem o.g. vorgestellten Programm \texttt{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 \texttt{Ctrl-t} oder \texttt{F10} öffnen Sie die Menüleiste.
2. Dort wählen Sie den Eintrag \textit{Ansichten} \rightarrow \textit{Neue einfache Paketansicht} aus.
3. Darin schränken Sie nun die Ansicht auf die installierten Pakete ein. Dazu drücken Sie \texttt{l} (für „limit“) und geben als Filter \texttt{-i} für „nur installierte Pakete“ ein. Den Vorgang schließen Sie mit der Eingabetaste ab.

Im Terminal finden Sie die zehn Pakete mit dem meisten Plattenplatzverbrauch durch eine Kombination von \texttt{aptitude} und dem Standard-UNIX-Kommando \texttt{tail}. An \texttt{aptitude} übergeben Sie dabei neben dem Unterkommando \texttt{search} mehrere Optionen. Während \texttt{--sort installsize} für die Sortierung nach dem Paketattribut „belegter Plattenplatzverbrauch“ sorgt, filtert \texttt{-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 \texttt{tail} ohne weitere Optionen beschränkt die Darstellung auf die letzten zehn Zeilen der Ausgabe von \texttt{aptitude} und somit die zehn größten Pakete.

\textbf{Suche nach den größten installierten Paketen mit \texttt{aptitude} (Namensliste)}

\begin{verbatim}
$ aptitude search --sort installsize '-i' | tail
  i A cube2-data - demo game and content for the Cube2 engine
  i A nuxviz-data - Nuxviz 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 al
  i A flightgear-data-al - FlightGear Flight Simulator -- standard AI
  i A nuxviz-textures - Textures for Nuxviz
  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
$
\end{verbatim}
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 10.8).

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  nexus-data
302 MB  torcs-data-tracks
306 MB  supertuxkart-data
320 MB  flightgear-data-aircrafts
466 MB  flightgear-data-ai
523 MB  nexus-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` und `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  nexus-textures
466 MB  flightgear-data-ai
$```

### 8.17 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.


**Ausgabe von `aptitude` zur Klärung der Installation (Variante 1)**
Für eine wesentlich ausführlichere Ausgabe nutzen Sie den Schalter \texttt{-v}. Dieser besitzt die übliche Langform \texttt{--verbose} und ergänzt die Ausgabe um alle Pakete, die zur Installation führen können. Hintergrund dafür ist, dass \texttt{aptitude} aus den Paketinformationen sämtliche Abhängigkeits- und Installationspfade errechnet und Ihnen diese auflistet.

**Ausführliche Darstellung der Installationspfade (Ausschnitt)**

$ aptitude --verbose why xpdf

\begin{verbatim}
  i  texlive-latex-extra Empfiehlt texpower (>= 0.2-2)
  i  A texpower Empfiehlt xpdf | pdf-viewer

  ...

  ...
  i  xpdf Liefert pdf-viewer

\end{verbatim}$

\begin{itemize}
  \item \texttt{texlive-latex-extra} Empfiehlt \texttt{texpower (>= 0.2-2)}
  \item \texttt{A texpower} Empfiehlt \texttt{xpdf | pdf-viewer}
  \item \texttt{xpdf Liefert pdf-viewer}
\end{itemize}

In Kombination mit dem Schalter \texttt{--show-summary} fassen Sie diese Informationen zusammen und erhalten eine kompaktere Darstellung. Der Schalter erlaubt die fünf Werte \texttt{no-summary}, \texttt{first-package}, \texttt{first-package-and-type}, \texttt{all-packages} und \texttt{all-packages-with-dep-versions}. Geben Sie den Schalter beim Aufruf nicht an, wird \texttt{show-summary} auf den Wert \texttt{no-summary} gesetzt. Spezifizieren Sie keinen Wert für \texttt{show-summary}, wird hingegen \texttt{first-package} angenommen. Damit erhalten Sie bei einem Aufruf die folgende Ausgabe – hier beispielhaft für das Paket \texttt{foomatic-db}:

**Zusammenfassung für das Paket foomatic-db**

$ aptitude --show-summary why foomatic-db

\begin{verbatim}
  Pakete, die foomatic-db erfordern:
  \begin{itemize}
    \item bluetooth
    \item cups-pdf
    \item printer-driver-gutenprint
  \end{itemize}
\end{verbatim}$

\begin{itemize}
  \item \texttt{bluetooth}
  \item \texttt{cups-pdf}
  \item \texttt{printer-driver-gutenprint}
\end{itemize}

Der Wert \texttt{first-package-and-type} liefert Ihnen zudem eine Begründung für die Abhängigkeit:

**Zusammenfassung für das Paket foomatic-db (mit Begründung)**

$ aptitude --show-summary=first-package-and-type why foomatic-db

\begin{verbatim}
  Pakete, die foomatic-db erfordern:
  \begin{itemize}
    \item [Hängt ab von] bluetooth
    \item [Hängt ab von] cups-pdf
    \item [Hängt ab von] printer-driver-gutenprint
  \end{itemize}
\end{verbatim}$

\begin{itemize}
  \item \texttt{[Hängt ab von] bluetooth}
  \item \texttt{[Hängt ab von] cups-pdf}
  \item \texttt{[Hängt ab von] printer-driver-gutenprint}
\end{itemize}

Eine Übersicht zu allen Paketpfaden, die zum genannten Paket führen, erhalten Sie mit Hilfe des Wertes \texttt{all-packages}. Dabei stehen die Buchstaben hinter den Paketnamen für \texttt{Recommends (R)}, \texttt{Conflicts (C)}, \texttt{Depends (D)}, \texttt{Replaces (?)} und \texttt{Provides (P)}.

**Zusammenfassung für das Paket foomatic-db (ausführlicher)**

$ aptitude --show-summary=all-packages why foomatic-db

\begin{verbatim}
  Pakete, die foomatic-db erfordern:
  \begin{itemize}
    \item 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
  \end{itemize}
\end{verbatim}$

\begin{itemize}
  \item \texttt{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
\end{itemize}
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)

```bash
$ 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
db
printer-driver-gutenprint H: cups E: foomatic-filters E: foomatic-db-engine 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)

```bash
$ aptitude why libruby-extras
Kann keinen Grund für die Installation von libruby-extras finden.
```


Ausgabe von aptitude zur Klärung der Installation (Variante 3)

```bash
$ 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)

```bash
$ aptitude why-not kdm
Kann keinen Grund für die Entfernung von kdm finden.
```

8.18 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. Sowohl dpkg, als auch apt lassen Sie die Aktionen nachverfolgen.
8.18.1 Statusdaten von dpkg


Recherche in den Logdateien von dpkg mittels grep

```bash
$ grep " install " /var/log/dpkg.log
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
```


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

```bash
$ 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 15:59:34 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

```bash
$ 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 15:59:34 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
```

⁴Ist die alternative zcat-Implementation aus dem Paket zutils installiert, ist die Option -f nicht mehr erforderlich.
8.18.2 Statusdaten von apt


Installation des Paketes apt-rdepends

| Start-Date: | 2016-07-20 16:00:48 |
| Commandline: | apt-get install apt-rdepends |
| Install: | apt-rdepends:amd64 (1.3.0-3) |
| End-Date: | 2016-07-20 16:00:49 |

Nehmen Sie eine Aktualisierung der Software vor, resultiert das bspw. in folgendem Eintrag:

Aktualisierung des Paketes wireshark samt dessen Abhängigkeiten

| Start-Date: | 2016-07-08 15:41:42 |
| Upgrade: | wireshark-common:amd64 (1.12.1+g01b65bf-4+deb8u6, 1.12.1+g01b65bf-4+deb8u7), wireshark-qt:amd64 (1.12.1+g01b65bf-4+deb8u6, 1.12.1+g01b65bf-4+deb8u7), wireshark:amd64 (1.12.1+g01b65bf-4+deb8u6, 1.12.1+g01b65bf-4+deb8u7) |
| End-Date: | 2016-07-08 15:41:48 |


8.19 Paketabhängigkeiten anzeigen


Vor der Veränderung des Paketbestands – d.h. dem Hinzufügen, Entfernen und Aktualisieren eines oder mehrerer Pakete – prüfen APT und aptitude, ob diese Paketabhängigkeiten nach den Änderungen des Bestand auf ihrem System weiterhin erfüllt sind. Die Abhängigkeiten müssen erfüllt sein, damit Ihr Linuxsystem weiterhin zuverlässig funktioniert und dieses für Sie benutzbar bleibt.


8.19.1 Die Abhängigkeiten eines Pakets auflisten

165 / 503

Unterschied zwischen **apt-cache** und **aptitude** bei der Suche nach Abhängigkeiten

Hier ist die Bedeutung der Unterkommandos bzw. Schalter zwischen den beiden Werkzeugen genau entgegengesetzt. Während **apt-cache depends** alle Abhängigkeiten eines gegebenen Pakets herausucht, „denkt“ **aptitude** andersherum und durchsucht die Abhängigkeiten aller Pakete nach einem Muster, sucht also nach Abhängigkeiten auf ein Paket bzw. einer Zeichenkette oder einem regulären Ausdruck. Daher entspricht **apt-cache depends** dem Aufruf **aptitude search** `~R` bzw. **aptitude search** `~dependence(...)` sowie **apt-cache rdepends** dem Aufruf **aptitude search** `~D` bzw. **aptitude search** `~dependends(...)`. 

8.19.1.1 Ausgabe der Paketabhängigkeiten mit dpkg-deb


Beachten Sie bitte beim Aufruf die etwas unintuitiven Abfolge der Parameter — erst den Schalter, danach den Dateinamen der deb-Datei und am Ende der Feldname. Für diese Information muss das entsprechende Paket nicht auf ihrem System installiert sein, sondern nur als Datei in einem Verzeichnis liegen und erreichbar sein.

**Ausgabe der Abhängigkeiten mittels dpkg-deb**

```
$ dpkg-deb -f xpdf_3.03-17+b1_amd64.deb Depends
libc6 (>= 2.4), libgcc1 (>= 1:4.1.1), libpoppler46 (>= 0.26.2), libstdc++6 (>= 4.1.1), libx11-6, libxm4 (>= 2.3.4), libxt6
```

8.19.1.2 Ausgabe der Paketabhängigkeiten mit apt-cache

**Variante 1** verfolgt einen typischen UNIX-Ansatz — es kombiniert die beiden Werkzeuge **apt-cache** und grep miteinander. Dazu filtert es die Ausgabe der vollständigen Paketinformationen, wie es **apt-cache** mit dem Unterkommando `show` liefert. Über eine Pipe und mittels grep filtert es danach die Zeile mit den Abhängigkeiten heraus, die mit dem Schlüsselwort `Depends` beginnt. Für diese Information muss das entsprechende Paket nicht auf ihrem System installiert sein, sondern nur in der Paketdatenbank als Paket gelistet sein.

**Suche mittels grep in der Ausgabe von apt-cache**

```
$ apt-cache show xpdf | grep Depends
Depends: libc6 (>= 2.4), libgcc1 (>= 1:4.1.1), libpoppler46 (>= 0.26.2), libstdc++6 (>= 4.1.1), libx11-6, libxm4 (>= 2.3.4), libxt6
```

**Variante 2** verwendet das spezifische Unterkommando `depends`. **apt-cache** gibt jedes benötigte Debianpaket in einer einzelnen Zeile in alphabetisch aufsteigender Reihenfolge und ohne die Versionsnummer aus. Für diese Information muss das entsprechende Paket nicht auf ihrem System installiert sein, aber in der Paketdatenbank gelistet sein.

**Detaillierte Ausgabe der Paketabhängigkeiten**

```
$ apt-cache depends xpdf
xpdf
 Hängt ab von: libc6
 Hängt ab von: libgcc1
 Hängt ab von: libpoppler46
 Hängt ab von: libstdc++6
 Hängt ab von: libx11-6
 Hängt ab von: libxm4
 Hängt ab von: libxt6
 Empfiehlt: poppler-utils
 poppler-utils:i386
```
Debian-Paketmanagement

Empfiehlt: poppler-data
Empfiehlt: gsfonts-x11
Empfiehlt: cups-bsd
  cups-bsd:i386
Kollidiert mit: <xpdf-common>
Kollidiert mit: <xpdf-common:i386>
Kollidiert mit: <xpdf-reader>
Kollidiert mit: <xpdf-reader:i386>
Ersetzt: <xpdf-common>
Ersetzt: <xpdf-common:i386>
Ersetzt: <xpdf-reader>
Ersetzt: <xpdf-reader:i386>
Kollidiert mit: xpdf:i386

Ohne weitere Optionen enthält die Ausgabe alle Abhängigkeiten, Empfehlungen und Konflikte 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 und pre-depends). Es entspricht damit exakt den Angaben im Feld Depends der Paketinformationen.

Beschränkung auf die wichtigen Abhängigkeiten mittels -i

```bash
$ apt-cache depends -i xpdf
xpdf
  Hängt ab von: libc6
  Hängt ab von: libgcc1
  Hängt ab von: libpoppler46
  Hängt ab von: libstdc++6
  Hängt ab von: libx11-6
  Hängt ab von: libxml2
  Hängt ab von: libxt6

$```

- **--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

**--recurse**

führt die Unterkommandos `depends` und `rdepends` rekursiv aus, so dass alle erwähnten Pakete einmal ausgegeben werden. Diese Liste kann sehr lang sein.

Die nachfolgende Ausgabe grenzt die Auflistung auf die Pakete ein, die lediglich als Vorschlag oder Empfehlung hinterlegt sind. Im Aufruf nutzen Sie dafür die Option `--no-depends`.

**Ausgabe der vorgeschlagenen und empfohlenen Pakete zu xpdf**

```bash
$ apt-cache depends xpdf --no-depends
xpdf
  Empfiehlt: poppler-utils
  poppler-utils:i386
  Empfiehlt: poppler-data
  Empfiehlt: gsfonts-x11
  Empfiehlt: cups-bsd
    cups-bsd:i386
  Kollidiert mit: <xpdf-common>
  Kollidiert mit: <xpdf-common:i386>
  Kollidiert mit: <xpdf-reader>
  Kollidiert mit: <xpdf-reader:i386>
  Ersetzt: <xpdf-common>
  Ersetzt: <xpdf-common:i386>
  Ersetzt: <xpdf-reader>
  Ersetzt: <xpdf-reader:i386>
  Kollidiert mit: xpdf:i386
```

**8.19.1.3 Recherche mit `apt-rdepends`**


**Ausgabe der Paketabhängigkeiten mit `apt-rdepends` (Ausschnitt)**

```bash
$ apt-rdepends xpdf | more
xpdf
  Depends: libc6 (>= 2.4)
  Depends: libgcc1 (>= 1:4.1.1)
  Depends: libpoppler46 (>= 0.26.2)
  Depends: libstdc++6 (>= 4.1.1)
  Depends: libx11-6
  Depends: libxm4 (>= 2.3.4)
  Depends: libxt6
  libc6
    Depends: libgcc1
  libgcc1
    Depends: gcc-4.9-base (= 4.9.2-10)
...`
```


**Erzeugung der Abhängigkeiten als `dot-Datei`**
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`

### 8.19.1.4 Ausgabe der Paketabhängigkeiten mit `aptitude`


#### Anzeige der Paketabhängigkeiten mit `aptitude`

```
$ aptitude search '~Rxpdf'
  i  libc6 - GNU C-Bibliothek: Dynamische Bibliotheken
  i  libgcc1 - GCC Support-Bibliothek
  i  A libpoppler46 - Bibliothek zur PDF-Darstellung
  i  libstdc++6 - GNU-Implementierung der Standard-C++-Bibliotheken
```
8.19.1.5 Ausgabe der Paketabhängigkeiten mit grep-status


Für diese Information muss das entsprechende Paket nicht auf ihrem System installiert sein, aber in der Paketdatenbank gelistet sein.

Im nachfolgenden Aufruf kommen -F Package zum Abgleich des Musters mit dem Paketnamen und -s Depends zur Ausgabe des Feldes für die Paketabhängigkeiten zum Einsatz.

Ausgabe der Paketabhängigkeiten mit grep-status

```
$ grep-status -F Package -s Depends xpdf
Depends: libc6 (>= 2.4), libgcc1 (>= 1:4.1.1), libpoppler46 (>= 0.26.2), libstdc++6 (>= 4.1.1), libx11-6, libxm4 (>= 2.3.4), libxt6
```


Ausgabe der Paketabhängigkeiten mit grep-aptavail

```
$ grep-aptavail -P -s Package,Depends,Architecture xpdf
Package: xpdf
Depends: libc6 (>= 2.4), libgcc1 (>= 1:4.1.1), libpoppler46 (>= 0.26.2), libstdc++6 (>= 4.1.1), libx11-6, libxm4 (>= 2.3.4), libxt6
Architecture: amd64

Package: xpdf
Depends: libc6 (>= 2.4), libgcc1 (>= 1:4.1.1), libpoppler46 (>= 0.26.2), libstdc++6 (>= 4.1.1), libx11-6, libxm4 (>= 2.3.4), libxt6
Architecture: i386
```

8.19.2 Anzeige der umgekehrten Paketabhängigkeiten


Unterschied zwischen apt-cache und aptitude bei der Suche nach Abhängigkeiten

8.19.2.1 Recherche mit `apt-cache`

Über das Unterkommando `rdepends` zeigt `apt-cache` alle Pakete an, die von weiteren Paketen abhängen, sind in der Ausgabe von `apt-cache` mit einem senkrechten Strich („Pipe“) gekennzeichnet. Für diese Information muss das entsprechende Paket nicht auf ihrem System installiert sein, aber in der Paketdatenbank gelistet sein.

Ausgabe der umgekehrten Paketabhängigkeiten mit `apt-cache` für das Paket `xpdf`

```
$ apt-cache rdepends xpdf
xpdf
 Reverse Depends:
  | xmds-doc
  | xfe
  | wiipdf
  | vim-latexsuite
  | python-scapy
  | ruby-tioga
  | python-tables-doc
  | page-crunch
  | octave-doc
  | muttprint-manual
  | mozplugger
  | mlpost
  | libmlpost-ocaml-dev
  | mc
  | libjgoodies-forms-java-doc
  | libinventor1
  | gprolog-doc
  | geomview
  | libfontconfig
  | eficas
  | auctex
```

8.19.2.2 Recherche mit `apt-rdepends`

`apt-rdepends` löst die Abhängigkeiten der Pakete zueinander noch weitaus stärker auf. Für diese Information muss das entsprechende Paket nicht auf ihrem System installiert sein, aber in der Paketdatenbank gelistet sein.

Ausgabe der umgekehrten Paketabhängigkeiten mit `apt-rdepends`

```
$ apt-rdepends -r xpdf
Reading package lists... Done
Building dependency tree
Reading state information... Done
xpdf
 Reverse Depends: eficas (6.4.0-1-1.1)
 Reverse Depends: muttprint-manual (0.73-5.1)
 Reverse Depends: wiipdf (1.4-2)
 eficas
 muttprint-manual
 wiipdf
```

8.19.2.3 Recherche mit aptitude


Um beispielsweise alle Pakete zu erhalten, die eine Abhängigkeit auf ein Paket mit der Zeichenkette `xpdf` im Paketnamen haben, nutzen Sie das Kommando `aptitude search '~Dxpdf'`. Das Ergebnis ist eine mehrspaltige Auflistung der Pakete mit deren Installationssstatus, Paketnamen und Kurzbeschreibung (siehe dazu „Liste der installierten Pakete anzeigen und deuten“ in Abschnitt 8.5).

Auflistung aller Pakete, deren Abhängigkeiten die Zeichenkette xpdf beinhalten, mit aptitude

```bash
$ 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
$```

Diese Liste beinhaltet auch das Paket `impressive`, welches eine Abhängigkeit auf `poppler-utils`, `mupdf-tools` und `xpdf-utils` hat. Will man nur Abhängigkeiten auf das Paket `xpdf`, so muß man Anfangs- und End-Anker verwenden:

Auflistung aller Pakete, die eine harte Abhängigkeit auf das Paket xpdf beinhalten, mit aptitude

```bash
$ aptitude search '~D^xpdf$'
p  eficas  - Graphical editor for Code Aster command files
p  muttprint-manual - Handbuch für muttprint
p  wiipdf - Präsentiert eine PDF-Datei mittels Wiimote
$```

8.19.3 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

```bash
# apt-get check
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
```

8.19.4 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

8.20 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.

Ergänzend stehen Ihnen mehrere Dienste für eine Recherche mittels Webbrowser zur Verfügung. Das umfaßt Dienste, die vom Debianprojekt unterhalten und gepflegt werden, aber auch kommerzielle Anbieter und private Initiativen.

8.20.1 Systemwerkzeuge

8.20.1.1 dpkg


**Fahnung nach den Paketen für xpdf mittels dpkg**

```
$ dpkg -l 'xpdf*'
```

```
++-==============-============-============-=================================
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).

### 8.20.1.2 APT und apt-cache


**Suche nach dem Begriff clint mittels apt-cache search**

```bash
$ apt-cache search --names-only clint
python-clint - Python Command-line Application Tools
python3-clint - Python Command-line Application Tools
```

Mehrere Suchbegriffe trennen Sie im Aufruf mittels Leerzeichen voneinander. Diese Begriffe werden dann mittels "und" als boolescher Ausdruck miteinander verbunden. Das Suchergebnis enthält dann nur Ergebnisse, in denen beide Suchbegriffe vorkommen. Das nächste Beispiel verwendet `lint` und `rpm` und listet nur das Paket `rpmlint` auf.

**Einfache Suche nach verfügbaren Paketen mittels apt-cache search und zwei Suchbegriffen**

```bash
$ apt-cache search --names-only lint rpm
rpmlint - RPM package checker
```

### 8.20.1.3 aptitude

Rufen Sie `aptitude` über die **Kommandozeile** auf, berücksichtigt es bei der Suche üblicherweise nur den Paketnamen. Es sucht jedoch auch auf Wunsch auch in der Paketliste und der Paketbeschreibung (siehe Abschnitt 8.21.2). 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 füllen Sie `aptitude` mit den folgenden Optionen:

- **~suchbegriff (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. Suche im Namen und der Beschreibung des Pakets.

Der Kommandozeilenaufruf von `aptitude` ist ähnlich zu `apt-cache` – auch hier folgt auf das Unterkommando `search` die Suchoption oder nur der Suchbegriff zur Recherche. `aptitude` interpretiert den Suchbegriff als Regulären Ausdruck im POSIX-Format.
Beispielhaft interessierte uns das Paket xpdf. 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 allen Paketen, in denen xpdf enthalten ist**

```
$ aptitude search xpdf
i xpdf      - Leseprogramm für das Portable Document Format (PDF)
p xpdf-reader - Übergangspaket für xpdf
p xpdf-utils
$ 
```

Wie oben genannt, grenzen Sie die Suche auf den exakten Paketnamen ein, indem Sie dem gesuchten Paketnamen ein `exact-name()` voranstellen. Damit findet aptitude nur noch ein einziges Paket:

**Suche nach dem exakten Paketnamen xpdf mittels aptitude**

```
$ aptitude search 'exact-name(xpdf)'
i xpdf      - Leseprogramm für das Portable Document Format (PDF)
$ 
```

Obiger Aufruf ist identisch mit `aptitude search ~n’ˆxpdf$’`. Der Suchbegriff ist hier als Regulärer Ausdruck angegeben und begrenzt das Textfragment auf die Zeichenkette xpdf, wobei mittels `ˆ` davor und `$` dahinter keine weiteren Zeichen erlaubt sind.

Beim Aufruf auf der Textoberfläche verhält sich aptitude genau entgegengesetzt zur Kommandozeile. Es zeigt nur die Paketnamen in der Auflistung an, die exakt mit dem angegebenen Suchmuster übereinstimmen. Die Auswahl der angezeigten Pakete begrenzen Sie mit der Taste l. In das Eingabefeld tragen Sie dann das gewünschte Suchmuster ein (siehe Abbildung 8.7 für den Eingabedialog).

```
Abbildung 8.7: Paketliste limitieren über die Aptitude-TUI
```

Nach aktuellem Entwicklungsstand erlaubt es die Textoberfläche von `aptitude` nicht, bei der Suche Reguläre Ausdrücke oder anderweitig Muster anzugeben. Das gelingt Ihnen nur über die Komandozeile.

### 8.20.1.4 grep-available und grep-status


### Verfügbar Pakete mittels `grep-available` 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
```


### Lediglich die installierten Pakete anzeigen, bei denen in der Beschreibung und im Paketnamen die Zeichenfolge `xpdf` enthalten ist

```
$ grep-status -F Description -P -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
```

Analog zu `grep` verfügt `grep-status` ebenfalls über den hilfreichen Schalter `-v` (Langversion `--invert-match`). Bei Bedarf verkehren Sie somit das Suchergebnis in das Gegenteil.

### 8.20.1.5 Synaptic

Bis zur Version 0.8 bot Ihnen Synaptic aus Abschnitt 6.4.1 zwei Varianten zur Suche an – einerseits eine Schnellsuche und andererseits eine ausführliche Suche. Die Schnellsuche verschwand mit der Version 0.8.


Abbildung 8.8: Ergebnis der Suche nach dem Fragment *fce* in Synaptic

### 8.20.1.6 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. Reguläre Ausdrücke bei der Formulierung des Suchstrings unterstützt es bislang nicht.

Abbildung 8.9: Ergebnis der Suche nach dem Fragment *top* in SmartPM

### 8.20.2 Browserbasierte Suche

#### 8.20.2.1 In Paketen blättern mittels *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 8.10). 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 8.10: Auswahl der Hilfeseiten zum Menüpunkt Netzwerkkommunikation

Wählen Sie ein Paket aus der Liste aus, erhalten Sie detailliertere Informationen dazu. Abbildung 8.11 zeigt das Vorgehen beispielhaft anhand des Pakets synaptic.

Neben dem Paketnamen versteht das Programm die folgenden beiden Schalter:

- **-s (Langform --stdout)**  
  die Ausgabe erfolgt nicht im Webbrowser, sondern auf dem Terminal.

- **-h Hostname**  
  Hostname des angefragten Rechners.


**Lokale Hilfedokumente zum Paket bash mittels dpkg-www anzeigen**

```
$ dpkg-www bash
```

Um die Informationen zu dem gleichen Paket zu erhalten, welches jedoch auf dem entfernten Rechner mit dem Hostnamen *kiste* installiert ist, funktioniert dieser Aufruf mit dem zusätzlichen Schalter -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 -s htop
```
$ 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/pixmaps
    /usr/share/pixmaps/htop.png
$ 

8.20.2.2 Suche über die Webseite des Debian-Projekts


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.


Aber nicht nur das — Sie können mit bestimmten Kurzformen der URL direkt die entsprechende Suche einleiten oder ein gewünschtes Paket anzeigen lassen:

**https://packages.debian.org/**

*suchbegriff* sucht nach Binärpaketen mit *suchbegriff* im Paketnamen, wobei der Paketname nicht mit dem Suchbegriff beginnen muß. Ebenso unterscheidet die Suche nicht zwischen Groß- und Kleinschreibung, sperrt sich aber gegenüber Regulären Ausdrücken. So sieht bspw. der Aufruf für Pakete mit *screen* im Namen aus:

https://packages.debian.org/screen

**https://packages.debian.org/src:**

*suchbegriff* sucht nach Quellpaketen mit *suchbegriff* im Paketnamen, z.B. so für Pakete mit *screen* im Namen:

https://packages.debian.org/src/screen

**https://packages.debian.org/release/paketname** zeigt die Informationen zum Binärpaket *paketname* in der Veröffentlichung *release* an, z.B. so für die Paketbeschreibung und die Abhängigkeiten des Paketes *screen* in der aktuellen, stabilen Debian-Veröffentlichung:

https://packages.debian.org/stable/screen
https://packages.debian.org/sprachkürzel/release/paketname
zeigt die Informationen zum Paket *paketname* in der Veröffentlichung *release* in der gewählten Sprache an, z.B. so für die Paketbeschreibung und die Abhängigkeiten des Paketes *screen* im aktuellen Stable-Release von Debian auf Deutsch:

https://packages.debian.org/de/stable/screen

https://packages.debian.org/source/release/sourcepaketname
zeigt die Informationen zum Quellpaket *sourcepaketname* in der Veröffentlichung *release* an, z.B. so für die Informationen und die zum Bau notwendigen Pakete des Quellpaketes *screen* in Debian 12 *Bookworm*:

https://packages.debian.org/source/bookworm/screen

Analog zu den Binärpaketen können Sie hier auch noch ein Sprachkürzel an den Anfang des Pfades setzen, um eine bestimmte Sprache zu erzwingen.

https://packages.debian.org/release/
zeigt alle Kategorien in der benannten Veröffentlichung (*release*) an, hier für Debian *unstable*:

https://packages.debian.org/unstable

https://packages.debian.org/release/kategorie
zeigt alle Binäerpakete in der entsprechenden Kategorie der gewählten Veröffentlichung (*release*) an (siehe „Sortierung der Pakete nach Verwendungszweck“ in Abschnitt 2.8), z.B. alle Binärpakete in der Kategorie *Mail* in der aktuellen, stabilen Veröffentlichung von Debian:

https://packages.debian.org/stable/mail/

Auch hier können Sie wieder ein Sprachkürzel an den Anfang des Pfades setzen, um eine bestimmte Sprache auszuwählen.

Anstelle des Namens einer Veröffentlichung — *bullseye*, *bookworm*, *sid*, etc. — kann auch stets ein Entwicklungsstand — *stable*, *testing*, *unstable*, etc. — verwendet werden.

8.20.2.3 Suche über die Webseite von Debian-Derivaten

Einige Derivate von Debian nutzen dieselbe Webanwendung zur Auflistung ihrer Pakete im Web. Den Autoren des Buches sind bisher bekannt:

Ubuntu (https://packages.ubuntu.com/)
unterstützt bisher keine Suite-Namen, denn es gibt bei Ubuntu bisher aber auch nur genau einen Suite-Namen namens *devel*. Der Aufruf für die Kategorie *mail* aus der Veröffentlichung *Xenial Xerus* in deutscher Sprache sieht wie folgt aus:

http://packages.ubuntu.com/de/xenial/mail/

Tanglu (http://packages.tanglu.org/)
unterstützt z.Zt. kein HTTPS. Daher erfolgt der Aufruf für die Kategorie *mail* aus der stabilen Veröffentlichung in deutscher Sprache wie folgt:

http://packages.tanglu.org/de/staging/mail/
Die für die Webseite des Debian-Projekts genannten Kurzformen sollten ebenfalls mit diesen Hostnamen funktionieren. Jedoch ist dabei zu beachten, dass andere Distributionen aufgrund anderer Release-Politiken ggf. keine Namen für Entwicklungsstände nutzen und damit auch diese Kurzformen nicht ermöglichen.


8.20.2.4 Suche über apt-browse.org


Die Suche erlaubt die gleichzeitige Recherche in mehreren Architekturen und in mehreren Veröffentlichungen. Eingepflegt sind bis dato die Veröffentlichungen Debian stable und unstable sowie die Ubuntu-Varianten der letzten 4 Jahre.


Abbildung 8.15: Ergebnis der Paketsuche nach htop über https://apt-browse.org/

8.20.2.5 Suche über apt-get.org

apt-get.org [apt-get.org] bietet Ihnen die Möglichkeit zur Recherche nach Paketen aus einem inoffiziellen Repository (siehe Abbildung 8.16). Das können neuere Paketversionen sein, aber auch Pakete, die noch nicht oder nicht mehr Bestandteil der Debian-Distribution sind.

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. Die Auswahl der Paketquelle sagt zudem nichts über die Qualität der darüber angebotenen Pakete aus. Auch wenn diese im Allgemeinen sehr hoch ist, bergen nicht verifizierbare Pakete ein gewisses Risiko.
Sehr hilfreich und zumeist auch der erste Anlaufpunkt 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 fundierte Paket einsortiert ist.

Abbildung 8.17 zeigt beispielhaft 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.
8.20.2.6 Rpmseek.com


Gefundene Pakete können Sie nicht nur inspizieren, sondern direkt von der angegebenen Quelle beziehen und installieren. Bitte beachten Sie aber, dass mit diesen Suchmaschinen gefundenene 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.


Integration distributionsfremder deb-Pakete

8.21 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 apt-cache aus dem Paket apt [Debian-Paket-apt], aptitude aus dem gleichnamigen Paket sowie grep-available und grep-status (beide aus dem Paket dctrl-tools [Debian-Paket-dctrl-tools]).

8.21.1 Suche mit apt-cache

Das Kommando apt-cache rufen Sie mit dem Unterkommando search und den gewünschten Suchbegriffen als Parameter auf. Es durchsucht daraufhin die Kurz- und Langbeschreibung des Pakets und gibt den Paketnamen samt einem Auszug aus der
Paketbeschreibung aus. Die Suche über den Paketnamen besprechen wir ausführlich in Abschnitt 8.20.1.2.

Der nachfolgende Aufruf demonstriert die Suche zum Stichwort lintian. Das Ergebnis ist eine zeilenweise Auflistung aus dem Paketnamen gefolgt von einem Auszug aus der Kurzbeschreibung zum Paket.

**Suche nach verfügbaren Paketen mittels apt-cache search und dem Suchbegriff lintian**

```
$ apt-cache search lintian
debaux - Debian-Hilfsprogramme
dput - Werkzeug für das Hochladen von Debian-Paketen
fixincludes - Repariert Header-Dateien, die nicht ANSI-kompatibel sind
gdeb1 - Einfaches Programm für Betrachtung und Installation von deb-Dateien - GNOME-GUI
lintian - Debian-Paketprüfung
devscripts - scripts to make the life of a Debian Package maintainer easier
eida - pbuilder mail interface
libconfig-model-dpkg-perl - editor for Dpkg source files with validation
libdebian-package-html-perl - generates HTML from a Debian source/binary package
pkg-perl-tools - collection of tools to aid packaging Perl modules in Debian
rpmlint - RPM package checker
```

Das Suchergebnis grenzen Sie ein, in dem Sie im Aufruf von `apt-cache` einen oder mehrere Suchbegriffe angeben. Hier sehen Sie obigen Aufruf mit den beiden Suchbegriffen `lintian` und `rpm`. Wie Sie sehen, reduziert sich die Liste der Suchtreffer auf zwei Pakete.

**Suche nach verfügbaren Paketen mittels apt-cache search und zwei Suchbegriffen**

```
$ apt-cache search lintian rpm
debaux - Debian-Hilfsprogramme
rpmlint - RPM package checker
```

Für ein noch ausführlicheres Suchergebnis kommt der Schalter `--full` (Langform `--full`) ins Spiel. Damit ist die Ausgabe identisch zum Aufruf `apt-cache search show` und gibt den vollständigen Datensatz für das Paket aus. Dieser umfaßt den Paketnamen, den Hashwert zum Paket sowie die hinterlegte Paketbeschreibung.

**Suche nach verfügbaren Paketen mittels apt-cache search und zwei Suchbegriffen (vollständige Paketbeschreibung)**

```
$ apt-cache search --full lintian rpm
Package: debaux
  Description-md5: fee7fd0fa25d42a9151a2e3b88577a50
  Description-de: Debian-Hilfsprogramme
    Dieses Paket enthält Perlprogramme und -module für die Erstellung und Veröffentlichung von Debian-Paketen.
    debaux-build lädt automatisch APT-Quellen vor dem Erzeugen herunter, fügt Patches und zusätzliche Quellen hinzu. Es kann die Pakete in einer chroot-Umgebung anlegen, die erzeugten Pakete werden lintian geprüft, auf dem lokalen Rechner installiert oder konvertiert nach RPM.
    debaux-build besitzt experimentell die Unterstützung für das downloaden von Perl-Modulen von CPAN und das Erzeugen passender Debian-Pakete.
    debaux-publish lädt die Pakete hoch und führt Skripte zum Erstellen der APT-Quellen und Paketdateien auf dem entfernten Rechner aus.
    debaux-publish unterstützt z.Zt. noch nicht die Pool-Struktur.

Package: rpmlint
  Description-md5: b8da9a736db7db144d0b4163fc42d180
  Description-en: RPM package checker
    rpmlint is a tool for checking common errors in rpm packages. rpmlint
```

```
can be used to test individual packages before uploading or to check an entire distribution. By default all applicable checks are performed but specific checks can be performed by using command line parameters.

rpmlint can check binary rpms (files and installed ones), source rpms, and plain specfiles, but all checks do not apply to all argument types. For best check coverage, run rpmlint on source rpms instead of plain specfiles, and installed binary rpms instead of uninstall binary rpm files.

The idea for rpmlint is from the lintian tool of the Debian project.

apt-cache bringt bislang keinen Schalter mit, um den Suchbegriff in der Ausgabe farblich hervorzuheben. Hier bleibt Ihnen nur die Erweiterung des Aufrufs mittels grep und dem dazugehörigen Schalter --color.

8.21.2 Suche mit Hilfe von aptitude

Ohne weitere Parameter im Aufruf gleicht aptitude den Suchbegriff nur mit den Paketnamen ab. Für die Suche über die Paketbeschreibung versteht es eine Reihe von Parametern:

~d{suchbegriff} (Langform ?description(suchbegriff))
Suche nach dem {suchbegriff} in der Paketbeschreibung.

?{term(suchbegriff)}
Volltextsuche nach {suchbegriff} im Namen und der Beschreibung des Pakets.

?{term-prefix(suchbegriff)}
Volltextsuche nach Begriffen, die den {suchbegriff} als Präfix beinhalten. Suche im Namen und der Beschreibung des Pakets.

Das nachfolgende Beispiel sucht nach allen Paketen, in deren Paketbeschreibung der Begriff 'lintian' vorkommt:

Suche in der Paketbeschreibung mittels aptitude

$ aptitude search ~dlintian
p debaux - Debian-Hilfsprogramme
i devscripts - scripts to make the life of a Debian Package maintainer
i A dput - Werkzeug für das Hochladen von Debian-Paketen
p elida - pbuilder mail interface
p fixincludes - Repariert Header-Dateien, die nicht ANSI-kompatibel sind
p gdebi - Einfaches Programm für Betrachtung und Installation von
p libconfig-model-dpkg-perl - editor for Dpkg source files with validation
p libdebian-package-html-perl - generates HTML from a Debian source/binary package
i A lintian - Debian-Paketprüfung
p pkg-perl-tools - collection of tools to aid packaging Perl modules in Deb
p rpmlint - RPM package checker

Die anderen beiden Filter liefern eine ähnliche Ausgabe wie oben, berücksichtigen jedoch sowohl den Paketnamen, als auch die Paketbeschreibung.

8.21.3 Suche mit grep-available und grep-status

Die beiden Werkzeuge grep-available und grep-status gehören zum Paket dctrl-tools [Debian-Paket-dctrl-tools]. Mit den Angaben -F Description (Beschreibungsfeld, Langform --field), -i (unabhängig von Groß- und Kleinschreibung,
Langform --ignore-case) sowie dem Suchbegriff als Parameter durchstöbert grep-available die gesamte Paketbeschreibung und liefert als Ergebnis den gesamten Datensatz zum gefundenen Paket zurück.


Verfügbare Pakete anzeigen, bei denen in der Beschreibung die Zeichenfolge deb enthalten ist

```
$ grep-available -F Description -i deb | grep Package | sort
Package: base-files
Package: base-passwd
Package: debconf
Package: debian-archive-keyring
Package: debianutils
Package: dpkg
Package: libapt-inst1.5
Package: libdebconfclient0
Package: tasksel
Package: tasksel-data
```


Lediglich die installierten Pakete anzeigen, bei denen in der Beschreibung die Zeichenfolge xpdf enthalten ist

```
$ grep-status -F Description -i -s Package xpdf | grep Package | sort
Package: libpoppler46
Package: libpoppler-cpp0
Package: libpoppler-glib8
Package: libpoppler-qt4-4
Package: poppler-utils
Package: xpdf
```

Analog zu grep verfügt grep-status ebenfalls über den hilfreichen Schalter -v (Langversion --invert-match). Bei Bedarf verkehren Sie mit diesem das Suchergebnis in das Gegenteil.

### 8.22 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 verantwortlichzeichnet. 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].

#### 8.22.1 Welche Pakete betreut ein Debian-Maintainer


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 ake@debian.org lautet der komplette Aufruf aptitude search ‘~m ake@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
Debian-Paketmanagement

$ 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 dilo            - 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 graphischen 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 Formatausgaben erklären wir Ihnen unter „aptitude Format Strings“ in Abschnitt 10.8. Eine Alternative bietet zudem das Aptitude Handbuch [aptitude-dokumentation-paketdarstellung].

Möchten Sie in der Ausgabe stattdessen 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' -m abe@debian.org'
aha             - Konvertiert ANSI-Farben nach HTML - Axel Beckert <abe@debian.org>
amora-applet    - use a bluetooth device as X remote control - Axel Beckert <abe@debian.org>
amora-cli       - use a bluetooth device as X remote control - Axel Beckert <abe@debian.org>
autossh         - SSH-Sitzungen und -Tunnel automatisch neu - Axel Beckert <abe@debian.org>
...
$

Das Programm grep-dctrl aus dem Paket dctrl-tools [Debian-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 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

Abbildung 8.19: Ergebnis der Suche nach dem Paketmaintainer in Synaptic

8.22.2 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`

```bash
$ dd-list screen xymon fping ack
Antti Järvinen <antti.jarvinen@katiska.org>
  screen (U)
Axel Beckert <abe@debian.org>
  ack (U)
  fping
  screen
  xymon (U)
Christoph Berg <myon@debian.org>
  xymon
Debian Perl Group <pkg-perl-maintainers@lists.alioth.debian.org>
  ack
```

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 ;-)
8.23 Paket nach Lizenz finden

Entwickeln Sie Software und stellen diese der Aussenwelt zur Verfügung — bspw. als Freie Software — gehört eine Softwarelizenz einfach dazu. Mit der Lizenz legen Sie fest, was jemand anderes mit ihrem Werk tun darf. Das betrifft sowohl die private Nutzung, als auch den kommerziellen Einsatz. Eine Mehrfachlizenzierung liegt vor, wenn Sie für beide Nutzungsarten jeweils eine unterschiedliche Lizenz angeben.


Behelfen können Sie sich bislang nur sofern, dass Sie die Paketverwaltung so einstellen, nur Pakete aus ausgewählten Distributionsbereichen zuzulassen. Möchten Sie beispielsweise ausschließlich Pakete auf ihrem System zulassen, die unter freien Lizenzstehen und auch nicht von unfreien Paketen abhängig sind (siehe „Distributionsbereiche“ in Abschnitt 2.9), begrenzen Sie diese auf den Bereich main (siehe „Die Datei /etc/apt/sources.list verstehen“ in Abschnitt 3.3).

8.24 Paket zu Datei finden


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.

8.24.1 Suche in bereits installierten Paketen


Suche nach dem Muster bsod, mittels dpkg

```
$ dpkg -S bsod.
xscreensaver-screensaver-bsod: /usr/share/applications/screensavers/bsod.desktop
xscreensaver-screensaver-bsod: /usr/share/man/man6/bsod.6x.gz
xscreensaver-screensaver-bsod: /usr/share/xscreensaver/config/bsod.xml
```


Suche nach der Pfadangabe /bsod.

```bash
$ dpkg -S /bsod.
```
$ dpkg -S /bsod.
dpkg-query: Kein Pfad gefunden, der auf Muster /bsod. passt
$ dlocate -S /bsod.
xscreensaver-screensaver-bsod: /usr/share/applications/screensavers/bsod.desktop
xscreensaver-screensaver-bsod: /usr/share/man/man6/bsod.6x.gz
xscreensaver-screensaver-bsod: /usr/share/xscreensaver/config/bsod.xml
$

8.24.2 Suche in noch nicht installierten Paketen


Liste der Paketinhaltlisten anzeigen (apt-file Version 3.0 aufwärts)

$ ls /var/lib/apt/lists/*Contents*
/var/lib/apt/lists/debian.ethz.ch_debian_dists_bullseye_contrib_Contents-amd64.lz4
/var/lib/apt/lists/debian.ethz.ch_debian_dists_bullseye_main_Contents-amd64.lz4
/var/lib/apt/lists/debian.ethz.ch_debian_dists_bullseye_non-free_Contents-amd64.lz4
/var/lib/apt/lists/debian.ethz.ch_debian_dists_bullseye-security_main_Contents-amd64.←
   diff_Index
/var/lib/apt/lists/debian.ethz.ch_debian_dists_bullseye-security_non-free_Contents-amd64.←
   lz4


search Suchmuster
Suche danach, in welchem Paket die als Suchmuster angegebene Datei enthalten ist. Ergebnis ist eine Liste aller Pakete, die das angegebene Suchmuster enthalten. apt-file sucht dabei nur nach Dateinamen, nicht jedoch nach Verzeichnism Parametern.

find Suchmuster
Alias für den Schalter search.

list Paketname
gibt den Paketinhalt aus, auf den das Suchmuster passt. Diese Aktion ist sehr ähnlich zum Aufruf dpkg -L, nur das hier die Pakete noch nicht installiert sein müssen.

show Paketname
Alias für den Schalter list.

-a (Langform --architecture)
Einschränkung der Suche auf die angegebene Architektur (siehe Abschnitt 1.2).

-D (Langform --from-deb)
Debian-Paketmanagement

- `f` (Langform: `--from-file`) sucht nach dem Vorkommen aller Suchmuster, die in der angegebenen Textdatei stehen.


- `l` (Langform: `--package-only`) Das Ergebnis ist nur der Paketname, auf den das Suchmuster passt. Dateinamen werden nicht berücksichtigt.


- `x` (Langform: `--regexp`) interpretiert das Suchmuster als Regulären Ausdruck, so wie ihn Perl versteht (PCRE). Ohne diesen Schalter fasst `apt-file` das Suchmuster als schlichte Zeichenkette auf.


**Aktuelle Strukturdatenbank**
Um mit `apt-file` arbeiten zu können, müssen nach der Installation des Paketes mindestens einmal die Paketinhaltslisten aktualisiert werden. Das nehmen Sie entweder mittels `apt-file update` oder, oder indem Sie die Paketlisten aktualisieren — z.B. mittels `apt update` oder `apt-get update`. **Bei Versionen von apt-file vor Version 3.0 geht dies noch nicht automatisch und nur mittels `apt-file update`.** Unterbleibt dieser Schritt, quittiert `apt-file` einen Aufruf zur Suche mit der Fehlermeldung »The cache is empty. You need to run "apt-file update" first.« (auf Deutsch: »Der Cache ist leer. Sie zuerst müssen "apt-file update" aufrufen.«)

Das nachfolgende Beispiel zeigt die Suche nach der Zeichenkette `fping`.

### Suche über die Strukturdatenbank mittels `apt-file`

```
$ apt-file search fping
cacti: /usr/share/cacti/site/scripts/ss_fping.php
fping: /usr/bin/fping
fping: /usr/bin/fping6
fping: /usr/share/fping/NEWS.Debian.gz
fping: /usr/share/fping/changelog.Debian.gz
fping: /usr/share/doc/fping/changelog.gz
fping: /usr/share/doc/fping/copyright
fping: /usr/share/lintian/overrides/fping
fping: /usr/share/man/man8/fping.8.gz
fping: /usr/share/man/man8/fping6.8.gz
icingsweb2-module-graphite: /usr/share/icingsweb2/modules/graphite/templates/fping.ini
monitoring-plugins-standard: /usr/lib/nagios/plugins/check_fping
monitoring-plugins-standard: /usr/share/monitoring-plugins/templates-standard/fping.cfg
netdata-core: /usr/lib/netdata/conf.d/fping.conf
netdata-plugins-bash: /usr/lib/netdata/conf.d/fping.conf
netdata-plugins-bash: /usr/lib/netdata/plugins.d/fping.plugin
python3-nova: /usr/lib/python3/dist-packages/nova/api/openstack/compute/fping.py
python3-nova: /usr/lib/python3/dist-packages/nova/tests/functional/api_sample_tests/ test_fping.py
smokeping: /usr/share/doc/smokeping/examples/config.fping-instances.gz

$```
8.24.3 Suche über die Webseite des Debian-Projekts

Die Webseite bietet ebenfalls eine Suche anhand einer Zeichenfolge an (siehe Abbildung 8.20). 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 8.20: Suche nach mupdf über die Webseite](image)

8.25 Paket auseinandernehmen

8.25.1 Mit ar in seine Bestandteile zerlegen

Manchmal kann es ganz nützlich sein, wenn Sie ein Debianpaket vorliegen haben und dieses deb-Datei in seine Bestandteile zerlegen können. Mit dem folgenden Aufruf über das Kommando ar gelingt Ihnen das wie folgt:

Ein Debianpaket auseinandernehmen

```bash
$ ar -xv atom-amd64.deb
x - debian-binary
x - control.tar.gz
x - data.tar.xz
$```

Der Schalter `-x` sorgt für das Auspacken und `-v` steht für `verbose`, d.h. ausführliche Ausgabe. Als Ergebnis erhalten Sie diese drei Dateien im aktuellen Verzeichnis:

debian-binary
- enthält die Versionsnummer des verwendeten Debian-Paketformats, beispielsweise "2.0".

control.tar.gz
- ist ein komprimiertes tar-Archiv mit den Metadaten des Pakets.

data.tar.gz, data.tar.xz, data.tar.bz2
- ist ein komprimiertes tar-Archiv mit den eigentlichen Dateien, die vom Programm benötigt werden.

ar ist so nett und erlaubt Ihnen auch nur das Auspacken einer einzelnen Komponente. Um bspw. lediglich die Versionsnummer zu erhalten, sieht ihr Aufruf dann wie folgt aus:

Eine Komponente des Debianpakets herausziehen
$ ar -xv atom-amd64.deb debian-binary
x - debian-binary
$

Weitere Informationen zu den verschiedenen Paketformaten und deren Komponenten entnehmen Sie bitte dem Abschnitt 4.2.

8.25.2 Mit dpkg die Installationsstruktur herausfinden

Im Gegensatz zu ar hilft Ihnen dpkg dabei, herauszufinden, in welche einzelnen Verzeichnisse die Inhalte eines Pakets kopiert werden. Der Schalter -L (Langform --listfiles) zeigt Ihnen lediglich den entsprechenden Pfad an (siehe Abschnitt 8.26). Mit Hilfe von -x (Langform --extract) packt dpkg das Paket hingegen vollständig aus. Intern reicht es dabei die Ausführung an das Werkzeug dpkg-deb weiter.

Als zusätzlichen Parameter geben Sie im Aufruf ein Verzeichnis an, in das die Inhalte aus dem Paket kopiert werden sollen. Das nachfolgende Beispiel nutzt . für das lokale Verzeichnis.

```
frank@mauritius:/projekte/metapackage/auspacken$ dpkg -x meta-mc_1.0_all.deb .
frank@mauritius:/projekte/metapackage/auspacken$ tree

meta-mc_1.0_all.deb
  usr
  share
    doc
      meta-mc
        changelog.gz
        copyright
        README Deb

4 directories, 4 files
frank@mauritius:/projekte/metapackage/auspacken$
```

Abbildung 8.22: Ein Paket mit dpkg -x auspacken

8.26 Paketinhalte anzeigen


Es sind mehrere Fälle zu unterscheiden, die jeweils unterschiedliche Aufrufe nachsichziehen:

- **das Paket ist bereits installiert**

- **das Paket ist nicht installiert, liegt aber als Datei vor**

- **das Paket muss nicht installiert sein, kann aber**
8.26.1 dpkg -L Paketname


Auflistung des Paketinhalts mit allen Pfaden via dpkg

```bash
$ dpkg -L xterm
./
/etc
/etc/X11
/etc/X11/app-defaults
/etc/X11/app-defaults/UXTerm-color
/etc/X11/app-defaults/UXTerm
/etc/X11/app-defaults/KOIBRXTerm-color
/etc/X11/app-defaults/KOIBRXTerm
/etc/X11/app-defaults/XTerm-color
/etc/X11/app-defaults/XTerm
...
$
```

8.26.2 dlocate -L Paketname


8.26.3 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

```bash
$ dlocate -ls xterm
drwxr-xr-x 22 root root 4096 Sep 11 09:12 /.
drwxr-xr-x 160 root root 12288 Feb 7 05:58 /etc/
drwxr-xr-x 13 root root 4096 Dez 30 2017 /etc/X11/
drwxr-xr-x 2 root root 4096 Jul 8 2020 /etc/X11/app-defaults/
-rw-r--r-- 1 root root 2400 Mär 1 2015 /etc/X11/app-defaults/KOIBRXTerm
-rw-r--r-- 1 root root 6217 Mär 1 2015 /etc/X11/app-defaults/KOIBRXTerm-color
-rw-r--r-- 1 root root 3609 Mär 1 2015 /etc/X11/app-defaults/XTerm
-rw-r--r-- 1 root root 6209 Mär 1 2015 /etc/X11/app-defaults/XTerm-color
-rw-r--r-- 1 root root 10201 Mär 1 2015 /etc/X11/app-defaults/XTerm
-rw-r--r-- 1 root root 6207 Mär 1 2015 /etc/X11/app-defaults/XTerm-color
...
$
```

8.26.4 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

```bash

```
Debian-Paketmanagement

8.26.5 apt-file show Paketname und apt-file list Paketname


Paketinhalt in kompakter Form mittels apt-file

```
$ apt-file show xterm
xterm: /etc/X11/app-defaults/KOI8RXTerm
xterm: /etc/X11/app-defaults/KOI8RXTerm-color
xterm: /etc/X11/app-defaults/UXTerm
xterm: /etc/X11/app-defaults/UXTerm-color
xterm: /etc/X11/app-defaults/XTerm-color
xterm: /usr/bin/koi8rxterm
xterm: /usr/bin/ltterm
xterm: /usr/bin/resize
xterm: /usr/bin/uxterm
xterm: /usr/bin/xterm
...
```

Unterschiedliches Verhalten von apt-file in den Veröffentlichungen

In Debian 8 Jessie bzw. vor Version 3.0 verhält sich apt-file show bzw. apt-file list anders als in den nachfolgenden Debian-Veröffentlichungen. Eine Suche mittels list und show gibt alle Pakete aus, in denen das Suchmuster im Paketnamen vorkommt. Eine Suche nach xterm liefert zum Beispiel auch Treffer für die Pakete ajaxterm, kxterm und xtermcontrol.

Ab Debian 9 Stretch bzw. apt-file Version 3.0 zeigen apt-file show und apt-file list nur noch die Dateien für genau das angegebene Paket an.

8.26.6 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 xterm illustrieren.

Ergebnis der Recherche zum Paket xterm
$ dglob -f xterm
/etc/X11/app-defaults/UXTerm-color
/etc/X11/app-defaults/UXTerm
/etc/X11/app-defaults/KO18RXTerm-color
/etc/X11/app-defaults/KO18RXTerm
/etc/X11/app-defaults/XTerm-color
/etc/X11/app-defaults/XTerm
/usr/share/man/man1/lxterm.1.gz
...
$


8.27 Nach Muster in einem Paket suchen


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.
...
$


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
8.28 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 8.26) 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
```


Binärdateien mit dem Namen `aptitude` mittels `whereis` anzeigen

```
$ whereis -b aptitude
aptitude: /usr/bin/aptitude /usr/bin/X11/aptitude /usr/share/aptitude
```

8.29 Manpages anzeigen


8.29.1 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 `apropos` 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 wie folgt aus:

**Verfügbare Manpages für das Paket aptitude mittels dlocate aufspüren**

```bash
$ dlocate -man aptitude
$ 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**

```bash
$ 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**

```bash
$ dlocate -lsman aptitude
/usr/share/man/cs/man8/aptitude-curses.8.gz
/usr/share/man/es/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/it/man8/aptitude-curses.8.gz
/usr/share/man/pl/man8/aptitude-curses.8.gz
/usr/share/man/de/man8/aptitude-curses.8.gz
$
```

### 8.29.2 Manpages aus noch nicht installierten Paketen anzeigen

Benötigen Sie Hilfe zu einem Paket, welches Sie nicht installiert haben, sind die beiden Werkzeuge `debman` und `debmany` aus dem Paket `debian-goodies` [Debian-Paket-debian-goodies] für Sie nützlich. `debman` zeigt Ihnen nur die Manpages zu einem Paket an, welches nicht installiert ist. Es unterstützt dabei diese beiden Schalter:

**-f Dateiname Name**

Anzeigen der Manpage `Name` zu einem bereits lokal vorliegenden deb-Paket `Dateiname`

**Aufruf der Manpage chase zur lokal vorliegenden Datei chase_0.5.2-4_amd64.deb**

```bash
$ debman -f chase_0.5.2-4_amd64.deb chase
```

**-p Paketname Name**

Herunterladen des Pakets `Paketname` mittels `debget` und Anzeigen der darin enthaltenen Manpage `Name`

**Herunterladen des Pakets chase und Anzeigen der darin enthaltenen Manpage chase**

```bash
$ debman -p chase chase
```
Das Werkzeug debmany funktioniert etwas anders. Es sammelt Manpages oder Dokumentation zu einem Paket zusammen, welches entweder bereits als Paket installiert ist, sich noch in einem Repository befindet oder als lokale deb-Datei vorliegt. Es wertet dazu die Paketdatenbank aus. Ergebnis ist ein Auswahlmenü wie in Abbildung 8.23, über das Sie die Betrachtung der gewünschten Dokumentation auswählen. Für das Debianpaket apt sieht der vereinfachte Aufruf wie folgt aus:

**Aufruf von debmany für das Paket apt**

```
$ debmany apt
```

![Abbildung 8.23: Aufruf von debmany für das Paket apt](image)

Wählen Sie einen Menüeintrag aus, öffnet sich ihr bevorzugtes Anzeigeprogramm dafür. debmany bietet Ihnen zudem mehrere Schalter an, über die Sie dieses Anzeigeprogramm angeben können:

- **-g**
  Sinnvoll für GNOME, das Anzeigeprogramm muß dazu .gz-Dateien unterstützen. Kurzform von `-m 'gnome-open man:%s'`.

- **-k**
  Sinnvoll für KDE bzw. Konqueror, das Anzeigeprogramm muß dazu .gz-Dateien unterstützen. Kurzform von `-m 'kfmclient exec man:%s'` bzw. `-m 'kfmclient exec %s'` für andere Dateien.

- **-m Programm**
  Bezeichnet das Anzeigeprogramm zur Darstellung der Manpages. Dieses muß dazu .gz-Dateien unterstützen.

- **-o Programm**
  Bezeichnet das Anzeigeprogramm zur Darstellung sonstiger Dokumentation im Verzeichnis `/usr/share/doc`.

- **-x**
  Sinnvoll für KDE, GNOME und XFCE, das Anzeigeprogramm muß dazu .gz-Dateien unterstützen. Kurzform von `-m 'xdg-open man:%s'`. xdg-open ist Bestandteil des Pakets xdg-utils.

Weiterhin stehen diese Schalter zur Verfügung, über die Sie das Verhalten von debmany steuern können:

- **-L Limit**
  Gibt die maximale Dateigröße der Dokumentation an, die heruntergeladen wird. Überschreitet die Datei das angegebene Limit, so werden Sie gefragt, ob der Download stattfinden soll. Ohne Angabe einer Einheit ist die Angabe in Bytes. Durch Anhängen der Buchstaben K, M, G oder T passen Sie die Einheit an.
-l

Sprachliste
Legt fest, in welchen zusätzlichen Sprachen außer Englisch die Manpages angezeigt werden. Sprachliste ist eine durch Komma getrennte Liste der Sprachkürzel, bspw. de, fr für deutsch- und französischsprachige Manpages. Die Angabe de* liefert alle Varianten, bspw. de_DE, de_AT und de_CH.

-z

Abbildung 8.24: Aufruf von debmany für das Paket apt mit Hilfe von Zenity

8.29.3 Suche über den Webbrowser
Abbildung 8.25: 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.

8.30 Konfigurationsdateien eines Pakets anzeigen

Die meisten Programme verfügen über Konfigurationsdateien, mit denen Sie das jeweilige Programm auf ihre Bedürfnisse individuell einstellen können. Um das tun zu können, hilft es Ihnen, zu wissen, welche Dateien das überhaupt sind und an welcher Stelle im Verzeichnisbaum sich diese Dateien befinden.


In fast allen Fällen sind in Paketen enthaltene Dateien, die unter /etc/ liegen, solche "Conffiles" während alle anderen Dateien aus Paketen typischerweise keine "Conffiles" sind. Den Autoren dieses Buches sind zumindest bisher keine Ausnahmen von diesen Regeln bewusst.

Es kommt vor, dass Pakete Dateien unter /etc/ erst bei der Installation anlegen. Das ist insbesondere dann der Fall, wenn deren Inhalt dynamisch generiert wird. Diese sind dann keine "Conffiles" im eigentlichen Sinne, werden aber teilweise von Programmen wie ucf [Debian-Paket-ucf] analog zu Conffiles behandelt. Wie dies gehandhabt wird, hängt vom jeweiligen Paket ab.

Um solche bei Installation oder Aktualisierung eines Paketes erzeugte Konfigurationsdateien aufzufinden, kommt man nicht umhin, direkt in die entsprechenden Verzeichnisse zu schauen. Meistens wird aber zumindest das Verzeichnis schon im Paket mitgeliefert.

Die sogenannten Conffiles sind leicht aufzufinden. Einerseits können Sie wie unter Abschnitt 8.26 beschrieben, alle Dateien eines Paketes auflisten lassen. Praktisch alle so aufgelisteten Dateien im Verzeichnis /etc/ und dessen Unterverzeichnisse sind
"Conffiles" und damit auch Konfigurationsdateien. Diese lassen sich typischerweise durch Anhängen von `grep /etc/` an den entsprechenden Befehlen herausfiltern.

Es gibt jedoch ein paar Programme, die diesen Schritt ohne Zuhilfenahme von `grep` durchführen können. Diese Programme wollen wir Ihnen im Folgenden vorstellen.


**Statusabfrage für das Paket bash mittels dpkg -s**

```
$ dpkg -s bash
Package: bash
Essential: yes
Status: install ok installed
Priority: required
Section: shells
Installed-Size: 6469
Maintainer: Matthias Klose <doko@debian.org>
Architecture: amd64
Multi-Arch: foreign
Version: 5.1-2
Replaces: bash-completion (<< 20060301-0), bash-doc (<= 2.05-1)
Depends: base-files (>= 2.1.12), debianutils (>= 2.15)
Pre-Depends: libc6 (>= 2.25), libtinfo6 (>= 6)
Recommends: bash-completion (>= 20060301-0)
Suggests: bash-doc
Conflicts: bash-completion (<< 20060301-0)
Conffiles:
  /etc/bash.bashrc 89269e1298b235f1b12b4c16e4065ad0d
  /etc/skel/.bash_logout 22bfb8c1dd9ab5f3813a2b25da67463f
  /etc/skel/.bashrc ee35a240758f374832e809ae0ea4883a
  /etc/skel/.profile f4e81ade7d6f9fb342541152d08e7a97
Description: GNU Bourne Again SHell
  Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. Bash also incorporates useful features from the Korn and C shells (ksh and csh).
  Bash is ultimately intended to be a conformant implementation of the IEEE POSIX Shell and Tools specification (IEEE Working Group 1003.2).
  The Programmable Completion Code, by Ian Macdonald, is now found in the bash-completion package.
Homepage: http://tiswww.case.edu/php/chet/bash/bashtop.html
```

Da das ein bißchen sehr viel Ausgabe ist im Vergleich zu dem, was uns interessiert, juckt es uns in den Fingern, doch wieder `grep` zu bemühen, um die Ausgabe auf das Relevante zu kürzen:

**Filtern nach Conffiles einer Statusabfrage mit dpkg -s**

```
$ dpkg -s bash | grep '^ /'
  /etc/bash.bashrc 89269e1298b235f1b12b4c16e4065ad0d
  /etc/skel/.bash_logout 22bfb8c1dd9ab5f3813a2b25da67463f
  /etc/skel/.bashrc ee35a240758f374832e809ae0ea4883a
  /etc/skel/.profile f4e81ade7d6f9fb342541152d08e7a97
```

Direkter zur gewünschten Information kommen Sie mit den beiden Programmen `cat` und `dlocate` aus dem gleichnamigen Paket [Debian-Paket-dlocate]. Mit `cat` lesen Sie die Paketdatenbank wie folgt aus (hier für das Paket `bash`):

**Anzeigen der Konfigurationsdateien über die Paketdatenbank**

```
$ cat /var/lib/dpkg/info/bash.conffiles
/etc/bash.bashrc
/etc/skel/.bash_logout
/etc/skel/.bashrc
/etc/skel/.profile
$ 
```

dlocate kennt dazu den Schalter `-conf` gefolgt vom Paketnamen. Wieder zeigen wir das für das Paket `bash`.

**Ermittlung der Conffiles zum Paket bash mittels dlocate**

```
$ dlocate -conf bash
/etc/bash.bashrc
/etc/skel/.bash_logout
/etc/skel/.bashrc
/etc/skel/.profile
$ 
```

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 bash mittels dlocate (ausführliche Ansicht)**

```
$ dlocate -lsconf bash
-rw-r--r-- 1 root root 1987 7. Dez 04:24 /etc/bash.bashrc
-rw-r--r-- 1 root root 220 2. Dez 2015 /etc/skel/.bash_logout
-rw-r--r-- 1 root root 3526 2. Dez 2015 /etc/skel/.bashrc
-rw-r--r-- 1 root root 807 6. Feb 2018 /etc/skel/.profile
$ 
```

Nachteil von dlocate: Es muss installiert sein und seine Datenbank muß dazu aktuell sein. In der Standard-Einstellung wird die Datenbank aber nur einmal pro Tag aktualisiert und funktioniert daher typischerweise nicht für frisch installierte Pakete.


**Anzeige der Conffiles des Paketes bash und ihrer Unversehrtheit mittels debsums**

```
$ debsums -e bash
/etc/skel/.bash_logout       OK
/etc/bash.bashrc             FAILED
/etc/skel/.profile           OK
/etc/skel/.bashrc            OK
$ 
```

8.31 Paketänderungen nachlesen

8.31.1 Das Änderungsprotokoll beziehen

Für jedes Debianpaket existiert im entsprechenden Quellpaket (siehe Abschnitt 2.7.4) ein Protokoll mit den erfolgten Änderungen im Paket, ein sogenanntes Changelog (Datei `debian/changelog`). Das ist eine Textdatei mit einzelnen Einträgen (siehe
Debian-Paketmanagement

Daraus ersehen Sie überblicksweise, was die Entwickler im Vergleich zur vorherigen Veröffentlichung des Paketes verändert haben und welche Änderungen oder Debian-spezifischen Anpassungen der Maintainer im Vergleich zum Originalquellcode zusätzlich vorgenommen hat.


Beziehen der Changelog-Informationen zum Paket smartpm mittels apt-get

$ apt-get changelog smartpm
Holen:1 http://metadata.ftp-master.debian.org smart 1.4-2 Changelog [8.573 B]
Es wurden 8.573 B in 1 s geholt (4.754 B/s).
...
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
...
$

Bei aptitude sieht das wie folgt aus:

Die Paketänderungen zum Paket smartpm mit Hilfe von aptitude 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
...
$


Format der erfolgten Änderungen
Debian-Paketmanagement

Jeder dieser Blöcke beginnt stets mit dem Namen des Quellpakets, gefolgt von der Versionsnummer des Pakets, der Veröffentlichung (zumeist unstable) (siehe Abschnitt 2.10), sowie der Dringlichkeit oder dem Schweregrad der vorgenommenen Änderungen. Dieser Wert ist einer aus low, medium, high, emergency und critical. Darunter stehen die jeweiligen Änderungen als einfache Aufzählung.

8.31.2 Zwei Paketversionen miteinander vergleichen

Wie oben bereits genannt, beschreibt das Änderungsprotokoll lediglich in groben Zügen, welche Korrekturen ein Paket erfahren hat. Um die detaillierten Änderungen zwischen zwei (aufeinanderfolgenden) Paketversionen zu erfahren, bedarf es einer genaueren Inspektion.

Zwei Werkzeuge sind uns bislang bekannt — debdiff und diffoscope [Debian-Paket-diffoscope]. Das Erstgenannte stammt aus dem Paket debscripts [Debian-Paket-devscripts] stammt, das zweite gehört hingegen zu dem Projekt Reproducible Builds [ReproducibleBuilds]. Während das Paket debscripts für debdiff eine recht überschaubare Paketgröße besitzt (derzeit etwa 1 MB), ist diffoscope zwar nur rund 60 KB groß, zieht aber die gesamte Entwicklungsumgebung für Reproducible Builds als Paketabhängigkeit mit — das kann dann durchaus 2 GB Platz beanspruchen.

Mit beiden Werkzeugen erhalten Sie eine Auswertung darüber, welche Dateien oder Verzeichnisse aus einem Paket im Vergleich zur vorherigen Version entfernt wurden sowie welche hinzugekommen, umbenannt oder auch verschoben wurden und welche Besitz- und Ausführungsrechte sich ggf. noch geändert haben. Der nachfolgende Aufruf zeigt debdiff mit zwei Versionen des Pakets xpenguins über den folgenden Aufruf:

**Aufruf von debdiff für zwei Pakete**

```bash
$ debdiff xpenguins*
```

[The following lists of changes regard files as different if they have different names, permissions or owners.]

Files in second .deb but not in first
```
- ------------------------------------
- rw-r--r-- root/root /usr/share/doc/xpenguins/changelog.Debian.amd64.gz
```

Control files: lines which differ (wdiff format)
```
- -------------------------------
Installed-Size: [-1119-] [+1114+]
{+Source: xpenguins (2.2-10)+}
Version: [-2.2-10-] [+2.2-10+b1+]
```

Der Aufruf von diffoscope ist ähnlich, hier am Beispiel für zwei Pakete von cheese:

**Aufruf von diffoscope für zwei Pakete**

```bash
$ diffoscope cheese_3.14.1-1_amd64.deb cheese_3.14.1-2_amd64.deb
```

Nachfolgend sehen Sie die Ausgabe der entdeckten Änderungen. Es ist eine Art Baumstruktur, die durchaus länger werden kann. Daher zeigt das nachfolgende Bild nur einen Ausschnitt.
Abbildung 8.26: Von diffoscope fundene Änderungen (Ausschnitt)


8.32 Paket auf unerwünschte Veränderungen prüfen


Wir unterscheiden hier zwei Fälle:

beziehen eines Paketes und Prüfung auf Unversehrtheit der Übertragung

wurde das Paket korrekt zu Ihnen übertragen und ist die genutzte Quelle vertrauenswürdig. 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 auf dem Paketmirror oder dem Übertragungsweg stattgefunden haben. Diese Schritte bilden die Vertrauensbasis für die von Ihnen bezogene Software.

überprüfen, ob die Inhalte eines installierten Pakets verändert wurden

bestehen zwischen der Version vom Paketmirror und den installierten Dateien auf Ihrem System Unterschiede, und wenn ja, welche sind das.

8.32.1 Prüfung eines Paketes auf Unversehrtheit


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
Packages.bz2) sowie deren Hashsummen als MD5-, SHA1- und SHA256-Variante. Mit der digitalen Signatur der Release-
Datei und den darin enthaltenen 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 Da-
teien, die für diese Veröffentlichung zur Verfügung stehen – und deren Hash-Summen. Dies stellt wiederum sicher, dass die
Paketedateien 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.

Eine lokale Kopie der vertrauenswürdigen Schlüssel passend zur Veröffentlichung verwalten Sie mit dem Programm apt-key
aus dem Paket apt [Debian-Paket-apt]. Auf dessen Bedienung gehen wir unter Abschnitt 3.12.3 ein.

Zu dem Programm 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.

Debian signiert seine einzelnen Pakete nicht kryptographisch mittels dpkg-sig, sondern vertraut auf die signierte RELEASE-
Datei und die darin gespeicherten Hashsummen der Pakete.

8.32.1.1 Nur ein Einzelpaket kryptographisch prüfen

Die kryptographische Signatur eines einzelnen Paketes überprüfen Sie mit Hilfe des Werkzeugs dpkg-sig [Debian-Paket-dpkg-
sig]. Hat das Paket keine kryptographische Signatur, sieht die Ausgabe so aus:

**Paket ohne kryptographische Signatur**

```
$ dpkg-sig --verify /var/cache/apt/archives/mc-data_3%3a4.8.18-1_all.deb
Processing /var/cache/apt/archives/mc-data_3%3a4.8.18-1_all.deb...
NOSIG
```

Im positiven Fall sehen Sie das folgende Ergebnis, hier für das in Kapitel 22 erzeugte Metapaket meta-mc:

**Paket mit kryptographischer Signatur**

```
$ dpkg-sig --verify meta-mc_1.0_all.deb
Processing meta-mc_1.0_all.deb...
GOODSIG _gpgbuilder 35F8DF9C884E36AB974460AFCFA72978D431AC07 1573823436
```

Nun überprüfen Sie, ob ein Debianpaket selbst noch eine kryptographische Signatur enthält. Sie packen dieses aus und schauen
nach, ob darin eine Datei namens _gpgbuilder enthalten ist:

**Dateiliste eines Pakets mit kryptographischer Signatur**

```
$ ar vx meta-mc_1.0_all.deb
x - debian-binary
x - control.tar.gz
x - data.tar.xz
x - _gpgbuilder
```

Im vorliegenden Fall enthält die Datei _gpgbuilder folgendes Klartextzertifikat („clearsign GnuPG signature“):

**Enthaltenes Klartextzertifikat**

```
$ cat _gpgbuilder
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
Version: 4
```
8.32.1.2 Mehrere Pakete prüfen

Hier bietet sich ein Aufruf in einer for-Schleife an, bspw. wie folgt:

**Mehrere Pakete prüfen**

```bash
$ for paket in *.deb; do dpkg-sig --verify $paket; done
```

Als Ausgabe erhalten das Prüfergebnis paketweise untereinander.

8.32.2 Die Inhalte eines bereits installierten Paketes überprüfen

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 auch das Nachschauen, ob die lokalen Dateien eines bereits installierten Paket später 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? (Bei der Installation fragt Debian mittlerweile, ob ihre Anpassungen bestehen bleiben sollen.)
- 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], diffoscope [Debian-Paket-diffoscope] sowie 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.
8.32.2.1 MD5-Summen zur Erkennung von Änderungen

Während Debian bei der Verifizierung der bezogenen Pakete auch SHA1- und SHA256-Hashsummen zur kryptographischen Absicherung verwendet (siehe dazu Abschnitt 8.32.1), 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.


8.32.2.2 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 vollständigen 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
292b696a5b879f106b87c15073c245cd  usr/bin/htop
194b840f963e6bbf2922981a1a6195c2  usr/share/applications/htop.desktop
7557092070931cbcb0fb9a6d74575542  usr/share/doc/htop/AUTHORS
0c9303726b090f478b383ddo59b3265f  usr/share/doc/htop/README
3add8faa10448f27bb30385b37eb1423  usr/share/doc/htop/changelog.Debian.gz
84555fa6bc74568a8e8de2a18072d5b2  usr/share/doc/htop/changelog.gz
ee765b42989a83c9b04a179b35e59e1  usr/share/doc/htop/copyright
58a889c99141c29451c50bb51d514c6  usr/share/man/man1/htop.1.gz
f059e3f0159a5aeb761d41514a17310  usr/share/menu/htop
5bbd19dc6ccfa0a74866a92f5cba75c  usr/share/man/pixmaps/htop.png
$ 
```

8.32.2.3 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
$ 
```
8.32.2.4 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:**

```bash
$ 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**

```bash
$ 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 die Datei enthalten ist (siehe Abschnitt 8.24) und installieren dieses dann erneut (siehe Abschnitt 8.39).

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:**

```bash
$ debsums -e sudo
/etc/pam.d/sudo OK
/etc/sudoers OK
/etc/init.d/sudo OK
/etc/sudoers.d/README 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
$
```

### 8.32.2.5 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**

```bash
$ 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
$
```

### 8.33 Liste der zuletzt geänderten Abhängigkeiten


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
libc6: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.

### 8.34 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 8.35). 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`, `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 Webbrovers. 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 8.27).

![Abbildung 8.27: Bezug des Pakets bash-doc über den Webbrowser](image)


**Download des Pakets nano mittels debget**

```
# debget nano
(nano -> 2.2.6-3)
```

```
% Total  % Received % Xferd  Average Speed Time Time Time Current
   100 360k 100 360k 0 0 1124k 0 --:--:-- --:--:-- --:--:-- 1124k
```

debget kann jedoch keine Quellpakete beziehen, dafür steht Ihnen dann dget aus dem Paket devscripts [Debian-Paket-devscripts] zur Verfügung.

Bezug des Pakets bash-doc via aptitude und Speicherung im Paketcache

```bash
# 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 „Paketverwaltung ohne Internet“ in Kapitel 41). Damit stellen Sie vorab bereits alle Pakete zusammen, die Sie im Bedarfsfall benötigen und installieren diese dann aus dem Paketcache oder aus einem lokalen Verzeichnis, ohne auf eine bestehende Netzverbindung angewiesen zu sein.

8.35 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 benötigen apt bzw. apt-get die Option `--no-download` zum Unterkommando `install`. aptitude kennt den Schalter bislang leider nicht.

Nachfolgende Ausgabe zeigt das für apt-get anhand des Pakets `bash-doc`, welches bereits im Paketcache liegt.

**Installation des Pakets bash-doc via apt-get aus dem Paketcache**

```bash
# 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 ...
#
```

Bei obigem Aufruf kümmern sich `apt` und `apt-get` darum, dass alle Paketabhängigkeiten erfüllt sind. Geht es Ihnen hingegen nur um ein einziges Paket, was keine Abhängigkeitskonflikte besitzt, kann auch `dpkg` die Aufgabe übernehmen. Dazu braucht `dpkg` dann den Dateinamen des Pakets im Aufruf, bspw. so:

**Installation des Pakets bash-doc via dpkg aus dem Paketcache**

```bash
# dpkg -i /var/cache/apt/archives/bash-doc.deb
...
Paket vom Paketmirror beziehen und automatisch (sofort) installieren


8.36 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 9 `Stretch` sieht der Eintrag wie folgt aus:

```
deb-src http://ftp.de.debian.org/debian/ stretch 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 8.32.1) 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]
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
$`

```

**Bezug der beiden Pakete bash-doc und libbash-doc via dget**

```bash
# 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
```

### 8.37 Sourcepakete anzeigen


#### 8.37.1 apt-cache verwenden

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 Projektseite (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**

```bash
$ 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: fb9e0999deb84fd7ea95fa41d4bf43852 1112 htop_1.0.1-1.dsc d3b80d905a6b0f03f1389687087f901 384683 htop_1.0.1.orig.tar.gz 5952c54e87d6d47ad9d541764491796 9113 htop_1.0.1-1.diff.gz Checksums-Sha1: 3ce3eb973c4399fd24c578643790de158b39fe87e 1112 htop_1.0.1-1.dsc bad226cc887a2b7ea5042879ed18e067812d030e 384683 htop_1.0.1.orig.tar.gz
```
Debian-Paketmanagement

8.37.2 apt-show-source verwenden

Das Werkzeug listet jedes Paket auf, dessen Versionsnummer sich von der installierten unterscheidet. Nachfolgend sehen Sie das für das Paket libspice-server1.

Ausgabe der Informationen zum Sourcepaket zu libspice-server1

```
$ apt-show-source -p libspice-server1

Inst. Package (Version) | Newest Source Package (Version)
-------------------------|-----------------------------
libspice-server1 (0.12.5-1+deb8u8) | spice (0.12.5-1+deb8u7)
```

8.38 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 8.34.


Wir empfehlen Ihnen, dpkg nur im Ausnahmefall zu benutzen. Der Umgang mit dpkg bzw. das Wissen um die Bibliotheken dahinter (siehe Kapitel 5) zählt zu den 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.

8.38.1 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, wieweiel 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 Platz 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 Platz zusätzlich belegt sein.
Möchten Sie fortsetzen? [Y/n/?] n

Abbruch.


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 8.41. Damit stellen Sie sicher, dass Sie mit einer aktuellen Paketliste arbeiten und darüber nur Pakete auswählen, die
sich auf den neuesten Stand befinden. Sie verhindern damit insbesondere, dass Sie veraltete Varianten auf Ihr System einpflegen und reduzieren gleichzeitig den Aufwand bei einer später Aktualisierung.

8.38.2 Durchführung


**Aufruf von apt-get install mit dem Quantifizierungsoperator** *

```
# apt-get install 'aptitude-doc*'  
Paketlisten werden gelesen... Fertig  
Abhängigkeitsbaum wird aufgebaut.  
Statusinformationen werden eingelesen.... Fertig  
Die folgenden NEUEN Pakete werden installiert:  
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.  
...  
```


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.


8.38.3 Begutachtung

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.

### 8.38.4 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`.

- `--trivial-only`  
  Gegenstück zur Option `--assume-yes`. Damit werden alle Fragen automatisch mit „NO“ beantwortet und kritische Aktionen bleiben außen vor.

- `-d` (Langform `--download-only`)
  die Pakete werden nur heruntergeladen, jedoch nicht installiert (siehe Abschnitt 8.34).

- `--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 8.39).

- `--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.

### 8.38.5 Besonderheiten bei aptitude


### 8.38.6 Erweiterungen ab APT 1.1

Ab der Version 1.1 verfügt APT über interessante Erweiterungen. Diese bzw. eine spätere Version ist ab Debian 9 *Stretch* verfügbar. Damit 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 6.4.5) 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**

```bash
# apt install /tmp/foo.deb ./bar.deb
... #
```
8.39 Pakete erneut installieren


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.


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, cups).

8.39.1 Wiederinstallieren vollständig entfernter Pakete

Wurden Pakete mit der Option --purge vollständig entfernt (siehe dazu Abschnitt 8.43), 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.


8.39.2 Wiederinstallieren von Paketen mit vorhandenen Konfigurationsdateien

Auch dieser Fall funktioniert meist ganz unspektakulär und so wie Sie es erwarten. Neben den bereits oben erwähnten Stolperstel- len 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.
8.39.3 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 den Aufruf `apt-get install --reinstall`, `aptitude reinstall` oder auch `cupt reinstall`. Zusätzlich benötigen Sie im Aufruf den entsprechenden Paketnamen als Parameter.


8.39.4 Typische Stolperfälle bei Wiederinstallieren mehrerer Pakete


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 8.24). Damit ermitteln damit das dazugehörige Paket im Handumdrehen.


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.

8.40 Pakete konfigurieren

Im Normalfall liegen die Debianpakete entweder bereits fertig konfiguriert oder mit einer vorbereiteten Konfiguration auf dem Spiegelserver. Dafür zeichnet der Paketmaintainer verantwortlich. Mit verschiedenen Werkzeugen zeigen Sie diese Konfiguration an oder justieren das Paket neu.

8.40.1 Bestehende Konfiguration eines Pakets anzeigen


Ausgabe der Konfiguration eines Pakets — hier `tzdata`
In obiger Ausgabe erscheint vor jeder Zeile ein * für die Frage zur Konfiguration, die Ihnen als Benutzer bereits bei der Einrichtung des Pakets gestellt wurde. Das kommt beispielsweise dann vor, wenn Sie Debian auf Ihrem System installieren — das Paket tzdata ist ein grundlegender Bestandteil des Installationsprozesses. Es legt die verwendete Zeitzone Ihres Systems fest (tzdata kürzt time zone data ab).

8.40.2 Konfiguration für alle Pakete auslesen


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

```bash
# debconf-get-selections | less
# Standardwortliste des Systems:
# Choices: american (American English), deutsch (New German), Manuelle Einrichtung 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
... 
#
```


Auslesen der Konfiguration für den Installer (Ausschnitt)

```bash
# debconf-get-selections --installer
...
# Typ des drahtlosen Netzwerks (WLAN):
# Choices: Infrastruktur-Netzwerk (Managed), Ad-hoc-Netzwerk (Peer-to-Peer)
netcfg netcfg/wireless_adhocManaged select Infrastructure (Managed) network
# Änderungen auf die Platte schreiben und verschlüsselte Datenträger konfigurieren?
partman-crypto partman-crypto/confirm boolean false
... 
#
```

8.40.3 Bestehende Konfiguration anwenden


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**

```plaintext
... 
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**
Als nächstes 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:

- Die Konfigurationsdateien („Conffiles“) des Pakets werden entpackt.
- Die bereits bestehenden Konfigurationsdateien („Conffiles“) für das Paket werden gespeichert. Falls dabei etwas schief geht, können diese wiederhergestellt werden.
- Stellt das Paket ein Maintainer-Skript namens postinst bereit, wird dieses abgearbeitet.


### 8.40.4 Konfiguration mit dpkg-reconfigure erneut durchführen

Für diesen Schritt hilft Ihnen das Werkzeug dpkg-reconfigure aus dem Paket debconf [Debian-Paket-debconf] weiter. Es greift dazu auf debconf zurück, welches wiederum eine Datenbank mit den Konfigurationseinträgen Ihrer Pakete unter
/var/cache/debconf benutzt.

dpkg-reconfigure kennt diese Optionen zum Aufruf:

- **-f (Langform --frontend)**


![Abbildung 8.28: Konfiguration des Pakets debconf](image)
Debian-Paketmanagement

--default-priority
verwendet low als Schwellwert für die Fragen, die Ihnen für die Konfiguration des Pakets gestellt werden

--force
erzwingt eine Konfiguration des Pakets, selbst wenn sich dieses in einem inkonsistenten oder defekten Zustand befindet

--no-reload
verhindert das erneute Laden der Vorlagen. Nützlich insbesondere dann, wenn die Vorlage defekt ist

-p (Langform --priority)
setzt den Schwellwert für die Fragen, die Ihnen für die Konfiguration des Pakets gestellt werden

-u (Langform --unseen-only)
zeigt Ihnen die Fragen an, die ihnen noch nicht zur Konfiguration des Pakets gestellt wurden

Mit dem Aufruf dpkg-reconfigure Paketname konfigurieren Sie ein bereits installiertes Paket erneut. Wir zeigen Ihnen das anhand des Pakets tzdata [Debian-Paket-tzdata], welches die Einstellungen für die Zeitzone steuert. Nach dem Aufruf zwei Dialogfenster — der Kontinent und die Stadt oder Region, die zu Ihrer Zeitzone passt (siehe Abbildung 8.29). Danach gibt Ihnen dpkg-reconfigure die aktuelle Zeitzone und die Uhrzeit aus — jeweils in lokaler Zeit (hier Mitteleuropäische Sommerzeit, kurz CEST) und in Universalzeit (Universal Time, kurz UTC).

Abbildung 8.29: Konfiguration des Pakets tzdata und regionale Auswahl

Einstellung der Zeitzone

```
# dpkg-reconfigure tzdata

Current default time zone: 'Europe/Berlin'
Local time is now: Sat Aug 6 00:46:08 CEST 2016.
```

8.41 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.

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 vier Unterkommandos zur Aktualisierung:

- update, verwendbar bei apt-get, apt und aptitude,
- install, 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 genannte Unterkommando beinhaltet eine spezifische Aktualisierung und wirkt sich entweder nur auf die Paketlisten (update) oder auf die Pakete selbst aus (install, upgrade, dist-upgrade, safe-upgrade und full-upgrade).

### 8.41.1 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 Sie in der Datei /etc/apt/sources.list bzw. dem Verzeichnis in /etc/apt/sources.list.d hinterlegt haben. Ausführlich gehen wir darauf unter „Liste der verfügbaren Pakete aktualisieren“ in Abschnitt 3.13 ein.

Mit diesem Unterkommando lösen Sie jedoch nicht die Aktualisierung der Pakete selbst aus — hierbei helfen Ihnen install, upgrade, dist-upgrade, safe-upgrade und full-upgrade.

### 8.41.2 install


Aktualisierung eines Paketes mittels apt-get install

```bash
# apt-get install ruby2.1
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden Pakete werden aktualisiert (Upgrade):
  ruby2.1
  1 aktualisiert, 0 neu installiert, 0 zu entfernen und 235 nicht aktualisiert.
Es müssen 277 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 0 B Plattenplatz zusätzlich benutzt.
```
Holen: 1 http://security.debian.org/ jessie/updates/main ruby2.1 amd64 2.1.5-2+deb8u6 [277 kB]
Es wurden 277 kB in 0 s geholt (354 kB/s).
Laden der Fehlerberichte ... Erledigt
»Found/Fixed«-Informationen werden ausgewertet ... Erledigt
Lese Changelog... Fertig
(Lese Datenbank ... 257112 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von .../ruby2.1_2.1.5-2+deb8u6_amd64.deb ...
Entpacken von ruby2.1 (2.1.5-2+deb8u6) über (2.1.5-2+deb8u5) ...
Trigger für man-db (2.7.0.2-5) werden verarbeitet ...
ruby2.1 (2.1.5-2+deb8u6) wird eingerichtet ...
#

Bei diesem Schritt werden die zu installierenden Softwarepakete zunächst im Paketcache unter /var/cache/apt/archives/ abgelegt. apt prüft dabei, ob für die heruntergeladenen Pakete auch genügend Speicherplatz zur Verfügung steht. Ist das nicht der Fall, beschwert es sich bei Ihnen mit nachfolgender Ausgabe:

**Fehlender Speicherplatz für den Paketcache**

1734 aktualisiert, 716 neu installiert, 29 zu entfernen und 0 nicht aktualisiert.
Es müssen 3.292 MB an Archiven heruntergeladen werden.
Nach dieser Operation werden 3.195 MB Plattenplatz zusätzlich benutzt.
E: Sie haben nicht genug Platz in /var/cache/apt/archives/.

Um die Situation zu bereinigen, räumen Sie vor der Aktualisierung der Pakete den Paketcache auf (siehe „Paketcache aufräumen“ in Abschnitt 7.5).

### 8.41.3 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 safe-upgrade** läßt hingegen das Installieren neuer Pakete zu. Ähnlich wie apt-get entfernt es dabei keine Pakete.


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


8.41.4 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.


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.

8.41.5 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:

3. Mit **U** merken Sie alle aktualisierbaren Pakete vor.
5. Mit **g** sehen Sie die Vorschau der anstehenden Aktionen an.
6. Drücken Sie nochmals **g**, um die vorbereiteten Aktionen auszuführen.

### 8.41.6 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**).

### 8.42 Pakete downgraden


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.


### 8.42.1 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]).

Wie oben schon benannt, sind diese Änderungen nicht immer rückwärtskompatibel und lösen Verwicklungen aus (Aktualisierungen sind eigentlich bereits hinreichend komplexe). 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 dazu Kapitel 20) oder das Übersetzen des Pakets aus den Quellen und das nachfolgende Einspielen des eigenen Binärpakets risikofreier ist.

### 8.42.2 Ablauf und Durchführung

#### 8.42.2.1 Bestehende Paketversionen klären


**Auflistung der verfügbaren Versionen zum Paket openvpn**

```bash
$ 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**

```bash
$ 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 benutzen Sie nachfolgend zum Downgrade.

Abbildung 8.30: Das Debian-Paketarchiv

8.42.2.2 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:

- Entfernen des derzeit installierten *openvpn*-Pakets mittels `apt-get remove openvpn`
- Einspielen des älteren *openvpn*-Pakets mittels `dpkg -ihv /var/cache/apt/archives/openvpn_2.3.4-5_amd64.deb`


8.42.2.3 Paket über die Angabe der Versionsnummer austauschen

APT akzeptiert als Parameter auch die explizite Angabe der Versionsnummer des Pakets. Falls das Paket noch nicht installiert ist, hilft dieser Aufruf:

```
apt-get install <package-name>=<package-version-number>
```

Ist das Paket jedoch schon installiert, gelingt dieser Aufruf:

```
apt-get install --reinstall <package-name>=<package-version-number>
```

In beiden Fällen ersetzen Sie `<package-name>` durch den tatsächlichen Namen des Paketes und `<package-version-number>` durch die gewünschte Versionsnummer. Für die Version 2.3.4-5 des Paketes *openvpn* sieht der Aufruf wie folgt aus:

```
Paket mit konkreter Versionsangabe (neu) installieren
# apt-get install --reinstall openvpn=2.3.4-5
```
8.42.2.4 Paket über die Angabe der Veröffentlichung austauschen

APT ist flexibel und erlaubt ebenfalls die Referenzierung eines Paketes über die explizite Angabe der Veröffentlichung. Dazu kommt der Schalter `-t` (Langform: `--target-release`) zum Einsatz:

```
apt-get -t=<target release> install <package-name>
```

Die Angabe `<target release>` benennt die Veröffentlichung, also bspw. `stable` oder `unstable`, aber auch den Namen wie `Bullseye` oder `Bookworm`. Für den Wert `<package-name>` geben Sie den tatsächlichen Namen des Paketes an. Für das Paket `openvpn` aus der vorherigen, stabilen Veröffentlichung (genannt `oldstable`) sieht der Aufruf dann wie folgt aus:

```
Paket mit Angabe der Veröffentlichung installieren

# apt-get -t=oldstable install openvpn
```

8.43 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 fünf 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. In Fall 5 zeigen wir Ihnen, wie sie Pakete für eine bestimmte Architektur entsorgen.

8.43.1 Fall 1: Paket einfach löschen


```
Löschens eines Pakets cssed mittels apt-get

# 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:

cssed
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

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.
```


Ersetzung durch ein alternatives Paket

ToDo: Beispiel mit Ersetzung durch alternatives Paket


8.43.2 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 von der Löschaktion 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 zuorden 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 1386  Akonadi PIM storage service
rc atop 1.26-2 1386  Monitor for system resources and process activity
rc audtty 0.1.12-1 1386  ncurses based frontend to audacious
...
```

8.44 Fall 3: Suche von Konfigurationsdateien änderter Pakete

Um das zu tun, bedarf es zunächst der Lokalisierung der Pakete, welche noch eine andere Konfiguration aufweisen. Dabei geht es nur um die Konfigurationsdateien (`conf files`), die sich unter dem Verzeichnis `/etc` befinden. Dateien in ihrem Homeverzeichnis bleiben von der Löschaktion 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 zuorden kann.

Suche nach ändern, 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 1386  Akonadi PIM storage service
rc atop 1.26-2 1386  Monitor for system resources and process activity
rc audtty 0.1.12-1 1386  ncurses based frontend to audacious
...
```

8.44 Fall 4: Suche von Konfigurationsdateien weiterer Pakete

Um das zu tun, bedarf es zunächst der Lokalisierung der Pakete, welche ebenfalls noch eine andere Konfiguration aufweisen. Dabei geht es nur um die Konfigurationsdateien (`conf files`), die sich unter dem Verzeichnis `/etc` befinden. Dateien in ihrem Homeverzeichnis bleiben von der Löschaktion 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 zuorden kann.
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**

```bash
$ aptitude search '~c'
c akonadi-backend-mysql - MySQL-Speicher-Backend für Akonadi
c akonadi-server - PIM-Speicherdienst Akonadi
c atop - Überwachung für Systemressourcen und Prozesse
c audtty - auf ncurses basierende Oberfläche für audat...```

8.43.3 **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, apt-get und aptitude unterstützen Sie mit der Kombination aus dem jeweiligen Namen des Werkzeugs und dem Unterkommando `purge` gefolgt vom Paketnamen, bspw. `apt purge mc` für die restlose Entfernung des Pakets `mc`.

**Löschen der Konfigurationsdateien mittels apt**

```bash
# apt purge mc
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden Pakete wurden automatisch installiert und werden nicht mehr benötigt:
  libconfuse-common libconfuse0
Verwenden Sie »apt-get autoremove«, um sie zu entfernen.
Die folgenden Pakete werden ENTFERNT:
  mc*
0 aktualisiert, 0 neu installiert, 1 zu entfernen und 108 nicht aktualisiert.
Nach dieser Operation werden 0 B Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/n]
(Leere Datenbank ... 253598 Dateien und Verzeichnisse sind derzeit installiert.)
Entfernen von mc (3:4.8.13-3) ...
Löschen der Konfigurationsdateien von mc (3:4.8.13-3) ...
```

In den Versionen vor 0.7.2 kennt apt-get das Unterkommando `purge` noch nicht. Ab der Version 0.7.7 (veröffentlicht 2007) funktionieren auch die Patches dazu zuverlässig. Falls Sie in die Situation kommen, mit einer Version vor 0.7.7 arbeiten zu müssen, benutzen Sie stattdessen im Aufruf die Kombination aus dem Unterkommando `remove` gefolgt von der Option `--purge` und den Namen der zu entfernenden Pakete. Nachfolgend sehen Sie das für apt-get und das zu entfernende Paket `cssed`.

**Löschen der Konfigurationsdateien mittels apt-get**

```bash
# 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]?
(Leere Datenbank ... 304031 Dateien und Verzeichnisse sind derzeit installiert.)
Entfernen von cssed ...
Löschen der Konfigurationsdateien von cssed ...
Trigger für menu werden verarbeitet ...
```

**Aufspüren noch konfigurierter Pakete**

```bash
# dpkg -l | egrep "^rc " | awk '{ print $2; }'
```

```bash
mediathekview
php5-mysqlnd
qgis-plugin-grass
samba
skype
subversion
svn-workbench
thunderbird
```

Um alle diese "Paketreste" in einem Rutsch aufzuräumen, kombinieren Sie diesen Aufruf bspw. mit `apt purge` wie folgt:

**Rückfrage vor dem Löschen aufgespürter, noch konfigurierter Pakete**

```bash
# apt purge $(dpkg -l | egrep "^rc " | awk '{ print $2; }')
```

```
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden Pakete werden ENTFERNT:
alsa-tools-gui* alsamixergui* bmon* bomber* cifs-utils* d2x-rebirth*
doxygen-gui* evince* firebird2.5-common* firebird2.5-server-common* geany*
geany-common* gedit* gtk+3* gkrellm* grass-core* gv* htdig* ibus* icedoc*
iceweasel1-i386* im-config* jed* jed-common* kdiffr* keyutils* ktorrent*
libapc10* libastro1* libatk1.0-0:i386* libavahi-compat-libdnssd1* libxcb-ares2*
libgphoto2:i386* libglu1-mesa* libglu1-mesa-gnu-gl* libglu1-mesa-shared*
libglvorbis1* libgstreamer-plugins-base-0.10* libgstreamer-plugins-good-0.10*
libgstreamer-plugins-good-0.10-3* libgnutls26* libgstreamer-plugins-ugly-0.10*
libgstreamer-plugins-ugly-0.10-3* libgstreamer1.0* libgstreamer-plugins-ugly1*
libgstreamer-plugins-ugly1-3* libgstreamer0.10* libgstreamer-plugins-ugly0.10*
libgstreamer-plugins-ugly0.10-3* libgstreamer0.10-3* libgstreamer-plugins-ugly10.0*
libgstreamer-plugins-ugly10.0-3* libgstreamer-plugins-ugly10.010*
libgstreamer-plugins-ugly10.010-3* libgstreamer-plugins-ugly100.0*
libgstreamer-plugins-ugly100.0-3* libgstreamer-plugins-ugly1000.0*
libgstreamer-plugins-ugly1000.0-3* libgstreamer-plugins-ugly10000.0*
libgstreamer-plugins-ugly10000.0-3* libgstreamer-plugins-ugly100000.0*
libgstreamer-plugins-ugly100000.0-3* libgstreamer-plugins-ugly1000000.0*
libgstreamer-plugins-ugly1000000.0-3* libgstreamer-plugins-ugly10000000.0*
libgstreamer-plugins-ugly10000000.0-3* libgstreamer-plugins-ugly100000000.0*
libgstreamer-plugins-ugly100000000.0-3* libgstreamer-plugins-ugly1000000000.0*
libgstreamer-plugins-ugly1000000000.0-3* libgstreamer-plugins-ugly10000000000.0*
libgstreamer-plugins-ugly10000000000.0-3* libgstreamer-plugins-ugly100000000000.0*
libgstreamer-plugins-ugly100000000000.0-3* libgstreamer-plugins-ugly1000000000000.0*
libgstreamer-plugins-ugly1000000000000.0-3* libgstreamer-plugins-ugly10000000000000.0*
libgstreamer-plugins-ugly1000000000000.0-3* libgstreamer-plugins-ugly100000000000000.0*
libgstreamer-plugins-ugly100000000000000.0-3* libgstreamer-plugins-ugly1000000000000000.0*
libgstreamer-plugins-ugly1000000000000000.0-3* libgstreamer-plugins-ugly10000000000000000.0*
libgstreamer-plugins-ugly10000000000000000.0-3*
```

0 aktualisiert, 0 neu installiert, 117 zu entfernen und 108 nicht aktualisiert.
Nach dieser Operation werden 0 B Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/N]...

...

**Paketweises Löschen aufgespürter, noch konfigurierter Pakete**

```bash
# for paket in $(dpkg -l | egrep '^rc ' | awk '{ print $2; }'); do apt purge $paket; done
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden Pakete werden ENTFERNT:
  asla-tools-gui*
0 aktualisiert, 0 neu installiert, 1 zu entfernen und 108 nicht aktualisiert.
Nach dieser Operation werden 0 B Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/n]...
```

---

### 8.43.4 Fall 4: Paket samt Konfigurationsdateien deinstallieren


**Löschen des Pakets cssed samt Konfigurationsdateien in einem Schritt**

```bash
# 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 fortfahren? [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 ...
```

---

### 8.43.5 Fall 5: Paket für eine ausgewählte Architektur entfernen

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). An den Paketnamen fügen Sie einen Doppelpunkt und den Namen der Architektur an. Um beispielsweise alle Pakete für die Architektur i386 vollständig von ihrem System zu entfernen, nutzen Sie diesen Aufruf:

**Vollständiges Entfernen aller installierten Pakete für die Architektur i386**

```bash
# apt-get remove --purge ".*:i386"
```
8.44 Umgang mit Waisen

Während der Verwendung von APT und `aptitude` werden die Abhängigkeiten der Pakete automatisch aufgelöst und daher auch zusätzlich benötigte Pakete eingespielt. Deinstallieren Sie zu einem späteren Zeitpunkt Pakete, werden bei `aptitude` alle nicht mehr benötigten Pakete ebenfalls wieder entfernt. APT (`apt, apt-get`) weist im Gegensatz dazu in der Standard-Einstellung nur auf nicht mehr benötigte Pakete hin, entfernt sie aber nicht automatisch.


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 8.44.2) und `deborphan` (siehe Abschnitt 8.44.3). Auf Ncurses basieren Orphaner und Editkeep (siehe Abschnitt 8.44.4) und auf GTK+ das Pendant Gtkorphan (siehe [Gtkorphan]). Darüber hinaus bietet `wajig` (Abschnitt 8.44.6) entsprechende Möglichkeiten zur Suche, die wir Ihnen ebenfalls nicht vorenthalten möchten.

8.44.1 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.


Bei `aptitude` ist das ganze Prozedere ein klein wenig anders. `aptitude` räumt in der Standard-Einstellung eigenständig auf. Das betrifft jedoch nur Pakete, die als "automatisch installiert" markiert sind und von denen wiederum kein manuell installiertes Paket abhängt (keine reverse dependencies bestehen).

Über die `aptitude::Delete-Unused` Option schalten Sie dieses Verhalten zu oder ab – entweder über die Benutzeroberfläche unter Optionen → Einstellungen, oder direkt in der Konfigurationsdatei von `aptitude`.

![Abbildung 8.32: Nicht verwendete Pakete automatisch entfernen in `aptitude`](image-url)
Um den Vorgang der Entfernung ungenutzter Pakete mit APT explizit anzustoßen, verfügen \texttt{apt} und \texttt{apt-get\`} über das Unterkommando \texttt{autoremove}. Seitdem nicht nur Aptitude sondern auch APT ein solche Funktionalität besitzt, hat die Bedeutung von \texttt{deborphan} und \texttt{debfoster} deutlich abgenommen.

Pakete automatisch entfernen mit dem Unterkommando \texttt{autoremove}

\begin{verbatim}
# 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.
#
\end{verbatim}

\texttt{aptitude} verfügt zudem über die Option \texttt{--purge-unused}, die noch einen Schritt weiter geht. Alle Pakete, die \texttt{aptitude} mangels Notwendigkeit entfernt, werden inklusive der dazugehörigen Konfigurationsdateien entsorgt. Diese Option können Sie über den Eintrag \texttt{aptitude::Purge-Unused} aktivieren.

\begin{itemize}
  \item \textbf{Verwendung der Option \texttt{--purge-unused}}
  \end{itemize}

Diese Option ist sehr mächtig und kombiniert eine ganze Reihe von Einzelschritten. Wir raten Ihnen daher, die Anwendung vorab genau zu prüfen.

8.44.2 debfoster

Das Paket \texttt{debfoster} [Debian-Paket-debfoster] ist ein Wrapper für die beiden Werkzeuge \texttt{dpkg} und APT. \texttt{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 \texttt{debfoster} das mit und fragt Sie, ob Sie das über die Abhängigkeit benannte Paket ebenfalls mit entfernen möchten.


Mit dem Kommando \texttt{debfoster -qv} erstellen Sie eine initiale Liste. Bei einem Folgeaufruf zeigt es Ihnen die Pakete, die die unerfüllten Abhängigkeiten aufweisen plus möglicherweise nicht mehr benötigte Pakete. \texttt{debfoster} warnt bei unerfüllten Abhängigkeiten (\texttt{warning}), wenn diese Pakete in der Liste der „zu behaltenden Pakete“ stehen.

\begin{verbatim}
# debfoster -qv

warning: package gnome-session-fallback: unsatisfied dependency on notification-daemon 0.7
warning: package gnome-session-fallback: forcing dependency on notification-daemon
warning: package tidymusic: 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
...
#
\end{verbatim}
Debian-Paketmanagement


- **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 8.19).

Ausgabe aller Abhängigkeiten mittels debfoster

```
# debfoster -d vim
Paket vim hängt ab von:
gcc-4.7-base libacl1 libattr1 libbinary 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 8.19).

Ausgabe aller umgekehrten Abhängigkeiten mit debfoster

```
# debfoster -e apt
Die folgenden 9 Pakete auf der Aufbewahrungsliste verlassen sich auf apt:
xara-gtk synaptic package-query 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:
abword acpi acpi-support anaconda apache2-utils apcalc apmd app-install-data apt-doc
apt-dpkg-ref apt-rdepends apvlv aqbanking-tools arora asciidoc ash aspell-de at ...
```
8.44.3 deborphan

Das Programm deborphan aus dem gleichnamigen Debian-Paket [Debian-Paket-deborphan] findet ungenutzte Pakete, die keine weiteren Abhängigkeiten zu anderen Paketen (siehe Abschnitt 8.19) aufweisen. Es gibt Ihnen eine Liste aller gefundenen Pakete aus, die Sie entfernen sollen, 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 Kurzschriftweise, in der Sie die Optionen in kompakter Form schreiben können. Der Aufruf deborphan -Pzs bewirkt das gleiche wie deborphan -P -z -s.

deborphan errät nicht mehr nützliche Pakete

```
# deborphan --guess-perl | sort
 gqview
 libchromaprint
 libconsole
 libcrypt-rc4-perl
 libgraphics-magick-perl
 libimage-exiftool-perl
 libbindicate-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
 dhc6dbd
 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

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 8.43.

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 eineiteils Tipparbeit und können darüber hinaus 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.
Möchten Sie fortfahren [J/n]?

#
```

**8.44.4 Orphaner und Editkeep**

`orphaner` und `editkeep` sind beides Benutzeroberflächen für `deborphan` (siehe Abschnitt 8.44.3) 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.


**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
```

...
8.44.5 **gtkorphan**

**gtkorphan** [Gtkorphan](#) ist ein graphisches Programm auf der Basis von GTK, welches *deborphan* (siehe Abschnitt 8.44.3) 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 Konfigurationdateien 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 8.35: gtkorphan bei der Arbeit

8.44.6 wajig


Die Analyse basiert auf dem Werkzeug deborphan (siehe Abschnitt 8.44.3). Daher muss das entsprechende Paket installiert sein, wenn Sie dieses Kommando verwenden möchten. Abbildung 8.36 zeigt das Ergebnis der Suche nach Waisen in der wajig-Shell.

Abbildung 8.36: wajig mit der Ausgabe des Kommandos orphans

8.44.7 rpmorphan

8.45 Paketoperationen erzwingen


8.45.1 Aktionen mit dpkg erzwingen


Die verschiedenen Schalter von dpkg zum Erzwingen einzelner Aktionen

$ dpkg --force-help

dpkg-Optionen zum Erzwingen – Verhalten steuern, wenn Probleme gefunden werden:
Warnen aber fortfahren: --force=<Ding>,<Ding>,...
Mit Fehler anhalten: --refuse=<Ding>,<Ding>,... | --no-force=<Ding>,...
Dinge erzwingen:
[*] all Alle Optionen zum Erzwingen setzen
[*] downgrade Paket durch eine niedrigere Version ersetzen
[!] configure-any Jedes Paket konfigurieren, das diesem helfen könnte
...[!] remove-reinstreq Pakete entfernen, die Installation erfordern
[!] remove-essential Ein essenzielles Paket entfernen

WARNUNG – Anwenden der mit [!] markierten Optionen kann Ihre Installation schwer beschädigen. Mit [*] markierte Optionen sind per Vorgabe aktiviert.

$ dpkg kennt zwei Wege — über den Schalter --force-Aktion sowie --no-force-Aktion und --refuse-Aktion. Bei ersterem warnt dpkg nur, setzt aber die ausgewählte Aktion fort. Bei den beiden letztgenannten hält es die Ausführung der Aktion an, sobald ein Fehler auftritt. Das sind die Schalter für die Aktionen im Einzelnen:

all
Alle Optionen zum Erzwingen setzen

architecture
Selbst Pakete mit falscher oder fehlender Architektur bearbeiten

bad-verify
Paket installieren, selbst wenn Authentizitätssüberprüfung misslingt

bad-path
Wichtige Programme nicht in PATH, Probleme wahrscheinlich

bad-version
Selbst Pakete mit fehlerhafter Version bearbeiten

breaks
Installieren, selbst wenn es andere Pakete beschädigt

confask
Anbieten, Konfigurationsdateien durch nicht neue Versionen zu ersetzen
confdef
Vorgabe für neue Konfigurationsdateien benutzen, wenn es eine gibt, nicht nachfragen. Wenn Vorgaben nicht gefunden werden können, nachfragen, außer confold oder confnew ist auch angegeben.

configure-any
Jedes Paket konfigurieren, das diesem helfen könnte

conflicts
Installation kollidierender Pakete erlauben

confmiss
Fehlende Konfigurationsdateien immer installieren

confnew
Immer neue Konfigurationsdateien verwenden, nicht nachfragen

confold
Immer alte Konfigurationsdateien verwenden, nicht nachfragen

hold
Nebensächliche Pakete bearbeiten, auch wenn auf halten gesetzt

depends
Alle Abhängigkeitsprobleme in Warnungen umwandeln

depends-version
Versionsabhängigkeitsprobleme in Warnungen umwandeln

not-root
Dinge versuchen zu (de)installieren, selbst wenn nicht root

overwrite
Datei eines anderen Pakets überschreiben

overwrite-dir
Verzeichnis eines Paketes mit Datei eines anderen überschreiben

overwrite-diverted
Umgeleitete Datei mit einer nicht umgeleiteten Version überschreiben

remove-reinstreq
Pakete entfernen, die Installation erfordern

remove-essential
Ein essenzielles Paket entfernen

script-chrootless
Nicht in die Umgebung von Betreuerskripten wechseln

unsafe-io
Beim Entpacken keine sicheren Ein/Ausgabe-Operationen durchführen

8.45.2 Aktionen mit \texttt{apt} erzwingen

- \texttt{apt}
  - Option \texttt{-f}

8.45.3 Aktionen an der Paketverwaltung vorbei

- \texttt{in /var/lib/dpkg/status} herumpfuschen
  - siehe \url{https://tipstricks.itmatrix.eu/force-aptitudeapt-get-ingoring-broken-dependencies/}

8.46 Paketstatusdatenbank reparieren

dpkg führt permanent Buch über alle Pakete auf einem Debian-System. Es merkt sich in seiner Paketstatusdatenbank den Zustand jedes einzelnen \texttt{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 \texttt{/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 \texttt{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.

8.46.1 Bit-Dreher reparieren

Die Einträge in der Paketstatusdatenbank sind 7-Bit-Werte, d.h. das achte Bit ist nicht gesetzt. Mittlerweile ist \texttt{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].


\textbf{Bit-Dreher in der Datei /var/lib/dpkg/status für das Paket \textit{geekcode} (Ausschnitt)}

\begin{verbatim}
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)
\end{verbatim}
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/


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

8.46.2 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.


 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.


8.46.3 Die Paketstatusdatenbanken von APT und aptitude


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).

8.47 Distribution aktualisieren (update und upgrade)

8.47.1 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 um 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.


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.

### 8.47.2 Vom **upgrade** zum **dist-upgrade**


### 8.47.3 Unsere empfohlene Reihenfolge

Wir empfehlen Ihnen, bei der Aktualisierung Ihrer Distribution die nachfolgenden Schritte nicht außer Acht zu lassen.


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 37.3.2).


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.


![](image.png)

Abbildung 8.38: Ausgabe während des Upgrades von Jessie auf Stretch

### 8.47.4 Anmerkungen


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.

9.1 Man- und Infopages


Benötigen Sie Hilfe für ein Paket, welches noch nicht installiert ist, helfen Ihnen die beiden Werkzeuge debman und debman weiter. Diese besprechen wir ausführlich in Abschnitt 8.29.2.

9.2 Dokumentation in /usr/share/doc/


Darüberhinaus werfen wir nachfolgend 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.

9.3 Die apt-dpkg-Referenzliste


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.


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.
9.5 Der Debian-Spickzettel

9.6 APT-Spickzettel von Nixcraft

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 B.

### 9.7 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 B.
9.8 Handbuch zu aptitude


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 damalige 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].

9.9 The Debian Administrator's Handbook


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 35). 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.

1das Werk basiert von jeher auf SuSE-Linux, reißt aber alle Bereiche des Linux-Alltags mit an
9.10 Weitere Bücher


Teil III

Praxis
Kapitel 10

APT und aptitude auf die eigenen Bedürfnisse anpassen


Recherchieren Sie nach der Konfiguration von APT, erhalten Sie viele Beiträge, die sich lediglich auf die Liste der Paketquellen konzentrieren. Diesem Thema gehen wir bereits ausführlich in Abschnitt 3.3 nach. Das nun folgende Kapitel konzentriert sich hingegen auf die weiteren Einstellungen von apt-get, apt und aptitude. Es soll Ihnen dabei helfen, das Verhalten dieser Werkzeuge auf ihre eigenen Bedürfnisse anzupassen, insbesondere wenn es um Spezialfälle geht.

Alle drei Werkzeuge greifen auf die gleichen Dateien zur Konfiguration zurück. APT liefert Erweiterungen — genannt Hooks (siehe Abschnitt 10.5), und aptitude erlaubt sowohl permanente, globale Einstellungen als auch lokale Einstellungen (siehe Abschnitt 10.6) sowie das Ändern der Verhaltens beim Aufruf von aptitude (siehe dazu Abschnitt 10.7).

10.1 Konfiguration von APT

APT und auch alle Programme, die die Bibliotheken von APT benutzen, greifen auf diverse Konfigurationsdateien zu. Über die Einstellungen darin werden unterschiedliche Funktionen abgebildet. Alle globalen Konfigurationsdateien befinden sich im Verzeichnis /etc/apt und beinhalten:

g /etc/apt

- **apt.conf** bzw. **apt.conf.d**
  Grundlegende Einstellungen von APT und den Bedienschnittstellen (Frontends). Siehe dazu Abschnitt 10.3.1.

- **listchanges.conf**

- **preferences** bzw. **preferences.d**
  Diese Einträge steuern, welche Versionen von Paketen zur Installation ausgewählt werden. Siehe dazu Abschnitt 10.3.2.

- **sources.list** bzw. **sources.list.d**
  Beinhaltet die Liste der Paketquellen, von denen APT Pakete bezieht. Ausführlich gehen wir darauf in Abschnitt 3.3 ein.

- **trusted.gpg** bzw. **trusted.gpg.d**

als einzelne Datei
   Das ist die ursprüngliche Variante der Konfigurationsdatei. Diese ist Ihnen als lokaler Systemadministrator vorbehalten.

als gleichnamiges Verzeichnis mit dem Suffix .d

Nachfolgend sehen beispielhaft den Inhalt des Verzeichnisses /etc/apt/sources.list.d/, wie er durchaus auf einer Desktopinstallation vorkommen kann. Enthalten sind die separaten Paketquellen für den Webbrowser Google Chrome sowie die beiden Messenger Signal und Skype.

Zusätzliche Paketquellen als separate Dateien

$ ls /etc/apt/sources.list.d/
google-chrome.list
signal-xenial.list
skype-stable.list
$

Jede aufgeführte Datei enthält eine oder mehrere, zusätzliche Paketquellen für das entsprechende Programm. Für den Messenger Signal sieht das bspw. wie folgt aus:

Zusätzliche Paketquelle für Signal (Release für Xenial)

$ cat /etc/apt/sources.list.d/signal-xenial.list
deb [arch=amd64] https://updates.signal.org/desktop/apt xenial main
$

Der Aufbau eines Eintrags entspricht der Datei /etc/apt/sources.list (siehe dazu Abschnitt 3.3).

10.2 Konfiguration von APT anzeigen


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 10.3).

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.*$;
APT::NeverAutoRemove:: ^kfreebsd-image.*$;
...
Nützliche Schalter und Optionen von `apt-config` sind:

**dump**
Nur der Inhalt des Konfigurationsbereichs wird angezeigt.

**shell**
Zugriff auf die Konfigurationsinformationen aus einem Shellskript heraus.

**-c (Langform --config-file)**
Angabe einer Konfigurationsdatei. Diese Datei wird zusätzlich zur üblichen Konfigurationsdatei gelesen.

**-h (Langform --help)**
Ausgabe der Hilfe zum Programm.

**-o (Langform --option)**
Angabe einer Option direkt beim Aufruf. Diesen Schalter können Sie mehrfach angeben.

**-v (Langform --version)**
Ausgabe der Version samt Architektur

**--empty**
Damit schließen Sie Optionen ein, die einen leeren Wert haben. Das ist die Standardeinstellung. Das Gegenstück dazu ist **--no-empty**.

**--format**
Darstellung der Ausgabe steuern. Zulässig sind diese Werte:

- `%f` — vollständiger, hierarchischer Name der Option
- `%n` — Zeilenumbruch
- `%N` — Tabulator
- `%t` — individueller Name der Option (Kurzform von `%f`)
- `%v` — Wert der Option

Die Standardeinstellung ist `%f "%v";%n. Die nachfolgende Ausgabe zeigt die Kurzform der Option inklusive deren Wert. Jede Zeile wird mit einem Semikolon abgeschlossen.

```
$ apt-config dump --format '%t "%v"; %n'
APT "";
Architecture "amd64";
Build-Essential "";
  "build-essential";
Install-Requires "1";
Install-Suggests "0";
Sandbox "";
...
$  
```

**--no-empty**
Damit schließen Sie Optionen aus, die einen leeren Wert haben. Das Gegenstück dazu ist **--empty**.

Was uns `apt-config` bislang nicht freiwillig verraten hat, ist, welche Option in welchem Konfigurationsschnipsel gesetzt wurde. Das ist bspw. dann wichtig, wenn Sie herausfinden müssen, welche Optionen sich überschreiben oder auch gegenseitig wieder aufheben. Für Debian 9 *Stretch* haben wir diese (provisorische) Lösung, um zumindest herauszubekommen, welche Konfigurationsdateien gelesen werden:

**Welche Dateien liest apt-config (Lösung für Debian 9 Stretch)**
$ strace -f -e open apt-config dump | grep "open(" ... 
open("/etc/apt/apt.conf.d/", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 4  
open("/etc/apt/apt.conf.d/00CDMountPoint", O_RDONLY) = 4  
open("/etc/apt/apt.conf.d/00trustcdrom", O_RDONLY) = 4  
open("/etc/apt/apt.conf.d/01autoremove", O_RDONLY) = 4  
open("/etc/apt/apt.conf.d/01autoremove-kernels", O_RDONLY) = 4  
open("/etc/apt/apt.conf.d/20listchanges", O_RDONLY) = 4  
open("/etc/apt/apt.conf.d/20packagekit", O_RDONLY) = 4  
open("/etc/apt/apt.conf.d/70debconf", O_RDONLY) = 4  
open("/var/lib/apt/lists/", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 4  
... 
$

Da sich in Debian 10 *Buster* der interne Aufruf von open() zu openat() geändert hat, sieht der Aufruf von apt-config nun wie folgt aus:

**Welche Dateien liest apt-config (Lösung für Debian 10 Buster)**

$ strace -f -e openat apt-config dump | grep "openat(" ... 
$

Nach unseren Tests funktioniert obige Lösung ebenfalls für Debian 11 *Bullseye* und 12 *Bookworm*.

### 10.3 Konfigurationsdateien von APT im Detail

#### 10.3.1 /etc/apt/apt.conf(.d) verstehen

In der Datei /etc/apt/apt.conf bzw. in den Dateien im Verzeichnis /etc/apt/apt.conf.d/ stellen Sie viele Aspekte von APT und anderen APT-Frontends ein. Alle diese Einstellungen überschreiben Sie bei Bedarf explizit auch auf der Kommandozeile (siehe Abschnitt 10.4). Darunter fallen u.a.:

- wie oft APT selbstständig Paketlisten und Pakete herunterladen soll (Schlüsselwort APT::Periodic),
- ob empfohlene (Recommends) oder vorgeschlagene Pakete (Suggests) per Default installiert werden sollen (Schlüsselworte APT::Install-Recommends und APT::Install-Suggests),
- welche Pakete nie automatisch entfernt werden sollen (Schlüsselwort APT::NeverAutoRemove),
- ob Paketlisten lokal entpackt oder komprimiert gespeichert werden sollen,
- welche Kompressionsalgorithmen beim Herunterladen von Paketlisten bevorzugt werden sollen (alle Schlüsselworte beginnend mit APT::Compressor),
- wohin APT Paketlisten und Pakete herunterladen soll. (Sollten Sie 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) (Schlüsselwort Dpkg::Progress-Fancy),
- 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 (siehe Abschnitt 10.5)
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 9 Stretch. Ob und welche Dateien tatsächlich vorhanden sind, hängt von den Werkzeugen ab, die Sie auf Ihrem System installiert haben.

**Beispielhafter Inhalt von /etc/apt/apt.conf.d/**

```bash
$ ls /etc/apt/apt.conf.d/
00CDMountPoint 01autoremove 20listchanges 70debconf
00trustcdrom 01autoremove-kernels 20packagekit
$
```

Die Datei 01autoremove-kernels beinhaltet bspw. Anweisungen, was mit älteren und inzwischen aktualisierten Kernel-paketen auf ihrem System passieren soll. Üblicherweise bleiben diese bei einer Aktualisierung des Pakets erhalten und werden von APT niemals automatisch entfernt. Der nachfolgende Ausschnitt demonstriert Ihnen das:

**Inhalt der Datei /etc/apt/apt.conf.d/01autoremove-kernels (Ausschnitt)**

```bash
// DO NOT EDIT! File autogenerated by /etc/kernel/postinst.d/apt-auto-removal
APT::NeverAutoRemove
{
    "^linux-image-3\.16\.0-4-amd64$";
    "^linux-image-4\.9\.0-11-amd64$";
    "^linux-headers-3\.16\.0-4-amd64$";
    "^linux-headers-4\.9\.0-11-amd64$";
    "^linux-image-extra-3\.16\.0-4-amd64$";
    "^linux-image-extra-4\.9\.0-11-amd64$";
    "^linux-signed-image-3\.16\.0-4-amd64$";
    "^linux-signed-image-4\.9\.0-11-amd64$";
    "^kfreebsd-image-3\.16\.0-4-amd64$";
    "^kfreebsd-image-4\.9\.0-11-amd64$";
    "^kfreebsd-headers-3\.16\.0-4-amd64$";
    "^kfreebsd-headers-4\.9\.0-11-amd64$";
    "^gnumach-image-3\.16\.0-4-amd64$";
    "^gnumach-image-4\.9\.0-11-amd64$";
    ".*-modules-3\.16\.0-4-amd64$";
    ".*-modules-4\.9\.0-11-amd64$";
    ".*-kernel-3\.16\.0-4-amd64$";
    ".*-kernel-4\.9\.0-11-amd64$";
    "^linux-backports-modules-.*-3\.16\.0-4-amd64$";
    "^linux-backports-modules-.*-4\.9\.0-11-amd64$";
    "^linux-tools-3\.16\.0-4-amd64$";
    "^linux-tools-4\.9\.0-11-amd64$";
};
```

**10.3.2 preferences bzw. preferences.d**

Hier handelt es sich um eine Datei bzw. ein Verzeichnis, in welchem Sie die Priorität eines oder mehrerer Pakete festlegen (siehe dazu Kapitel 20). Der Vorgang heißt Pinning und erlaubt es Ihnen, Pakete aus einer anderen Debian-Veröffentlichung zu verwenden — bspw. stable, testing oder unstable — ohne die Notwendigkeit, dabei gleich ihr ganzes System auf diese Veröffentlichung aktualisieren zu müssen.

Die Paketverwaltung respektiert ihre Festlegung und berücksichtigt das bei der Aktualisierung sowie bei den Paketabhängigkeiten. Genutzt wird ein Zahlenwert als Priorität — je höher dieser ist, umso mehr wird die gewählte Veröffentlichung bevorzugt.

Nachfolgendes Beispiel gibt Paketen aus testing eine höhere Priorität als den Paketen aus unstable:

**Veröffentlichungen priorisieren**
Die Datei preferences bzw. das Verzeichnis preferences.d ist ursprünglich leer. Eine detaillierte Beschreibung zu Pinning finden Sie bspw. in Abschnitt 20.4.

10.3.3 cron.daily/apt

• Begriff und Nutzen

10.4 Konfigurationsoptionen von APT

• Dazu verwenden Sie als Schalter -o, gefolgt von der Form <Schlüssel>=<Wert>.

10.5 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

10.6 Konfigurationsdateien von aptitude

aptitude verwendet alle Konfigurationsdateien von APT plus seine eigenen. Diese finden Sie in der Datei .aptitude/config in ihrem Home-Verzeichnis. Hier werden auch die interaktiven Änderungen der Konfiguration in der Textoberfläche von aptitude gespeichert. aptitude benutzt dabei die gleiche Syntax wie APT.

Um diese Einstellungen auf dem gesamten System anzuwenden, speichern Sie diese bspw. /etc/apt/apt.conf.d/lokale-aptitude. Diese Einträge 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 "";
```
10.7 aptitude: 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?

10.8 aptitude Format Strings

Mit Format Strings legen Sie die Ausgabe anhand von vorgegebenen Platzhaltern fest. 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].


<table>
<thead>
<tr>
<th>Platzhalter</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>%a</td>
<td>das Flag für die Aktion des Pakets (Action Flag)</td>
</tr>
<tr>
<td>%A</td>
<td>die ausführlichere Beschreibung des Flags der Aktion</td>
</tr>
<tr>
<td>%B</td>
<td>die Anzahl der kaputten Pakete (Broken Packages)</td>
</tr>
<tr>
<td>%c</td>
<td>der aktuelle Paketstatus (Current State Flag)</td>
</tr>
<tr>
<td>%C</td>
<td>eine ausführlichere Beschreibung des aktuellen Paketstatus</td>
</tr>
<tr>
<td>%d</td>
<td>die kurze Paketbeschreibung (Description)</td>
</tr>
<tr>
<td>%D</td>
<td>die Größe der Paketdatei</td>
</tr>
<tr>
<td>%E</td>
<td>der Name des Source-Pakets</td>
</tr>
<tr>
<td>%H</td>
<td>der Name des Rechners, auf dem aptitude gerade ausgeführt wird (Hostname)</td>
</tr>
<tr>
<td>%I</td>
<td>benennt die höchste Priorität, die einer Paketversion zugewiesen wurde</td>
</tr>
<tr>
<td>%i</td>
<td>gibt die Programmversion von aptitude aus</td>
</tr>
<tr>
<td>%m</td>
<td>der Name des Paketmaintainers (Maintainer)</td>
</tr>
<tr>
<td>%M</td>
<td>gesetzt, falls das Paket automatisch installiert wurde (Automatic Flag)</td>
</tr>
<tr>
<td>%n</td>
<td>gibt die Programmversion von aptitude aus</td>
</tr>
<tr>
<td>%N</td>
<td>gibt den Namen des Programms aus, i.d.R aptitude</td>
</tr>
<tr>
<td>%o</td>
<td>gibt eine Schätzung der Datenmenge zurück, die vom Repository bezogen wird</td>
</tr>
<tr>
<td>%p</td>
<td>der Paketname (Package Name)</td>
</tr>
<tr>
<td>%P</td>
<td>die Paketpriorität (Priority)</td>
</tr>
<tr>
<td>%r</td>
<td>gibt die geschätzte Anzahl Pakete an, die von diesem Paket abhängen (Reverse Depends Count)</td>
</tr>
<tr>
<td>%R</td>
<td>eine Abkürzung für die Priorität des Paketes (Abbreviated Priority)</td>
</tr>
<tr>
<td>%s</td>
<td>der Bereich, in den das Paket eingeordnet ist (Section)</td>
</tr>
<tr>
<td>%S</td>
<td>der Vertrauensstatus des Paketes; ist ü, falls es nicht aus einer vertrauenswürdigen Quelle stammt</td>
</tr>
<tr>
<td>%t</td>
<td>der Name des Archivs, aus dem das Paket stammt</td>
</tr>
<tr>
<td>%T</td>
<td>gibt einen * zurück, falls das Paket getagged wurde, ansonsten nichts</td>
</tr>
<tr>
<td>%u</td>
<td>falls die anstehenden Aktionen die Größe des verfügbaren Speicherplatzes auf dem Datenträger verändern, gibt dieser Platz eine entsprechende Mitteilung aus</td>
</tr>
<tr>
<td>%v</td>
<td>die aktuell installierte Version des Pakets, falls installiert (Version)</td>
</tr>
<tr>
<td>%V</td>
<td>gibt die Paketversion aus, die installiert würde, falls die Aktion Paket installieren ausgeführt wird (Candidate Version)</td>
</tr>
<tr>
<td>%Z</td>
<td>die Größe des Speicherplatzes, der zusätzlich benutzt oder freigegeben wird, wenn das Paket installiert, entfernt oder aktualisiert wird (Size Change)</td>
</tr>
</tbody>
</table>
Die Voreinstellung von aptitude beinhaltet die fünf Platzhalter `%c`, `%a`, `%M`, `%p` und `%d`. Es umfasst somit die einzelnen Spalten mit dem Paketstatus, der Aktion, das Automatic Flag, dem Paketnamen und der Paketbeschreibung.


**Individuelle Gestaltung des Ausgabeformats von aptitude bei der Suche nach den verfügbaren Paketen mit asciidoc im Namen**

```
$ aptitude search -F '%c %p' asciidoc
i asciidoc
v asciidoc:i386
i asciidoctor
i asciidoctor-doc
$.
```


**Installationsstatus, installierte Version und Paketname mit fester Breite**

```
$ aptitude search -F '%c | %20v | %p' asciidoc
i | 8.6.9-3 | asciidoc
v | <keine> | asciidoc:i386
i | 1.5.4-1-bpo8+1 | asciidoctor
i | 0.1.4-3 | asciidoctor-doc
$.
```


Das Ergebnis ist zunächst erstmal der komplette, übliche Ausgabestring aus Paketstatus, der Aktion, dem Automatic Flag, dem Paketnamen und der kurzen Paketbeschreibung. Um das noch auf den Paketnamen zu reduzieren, kommt der Format String %p ins Spiel.

**Lediglich nicht automatisch installierte Pakete auflisten**

```
$ aptitude search ‘~i !~M’ -F '%p'
abiword
abs-guide
ack-grep
acl
acpi
acpi-support-base
acpid
acpitool
adduser
adequate
...
$.
```
Die obige Liste beinhaltet auch alle essentiellen Pakete (siehe Abschnitt 2.13). Diese filtern Sie über die zusätzliche Angabe ‘!~E’ heraus. Damit ändert sich ihr Aufruf wie folgt:

**Nicht automatisch installierte und nicht essentielle Pakete auflisten**

```bash
$ aptitude search '!~i !~M' -F '%p'
abiword
abs-guide
ack-grep
...
$
```

**Anmerkung**


Geht es Ihnen jedoch darum, lediglich die Namen der Pakete auszugeben, die den Begriff *asciidoc* im Namen tragen und auch installiert sind, hilft aus unserer Sicht nur die Kombination aus `asciidoc` und `grep` wie folgt weiter:

**Alle installierten Pakete auflisten, die asciidoc im Namen tragen**

```bash
$ aptitude search '~i' -F '%p' | grep asciidoc
asciidoc
asciidoctor
asciidoctor-doc
$
```

Arbeiten Sie mit Repositorys vom Fremdanbietern, hängt die Zuverlässigkeit der Pakete vom Fremdanbieter ab. Mit dem nachfolgenden Aufruf — einer Kombination aus `aptitude` und `egrep` — finden Sie alle installierten Pakete, die aus nicht-vertrauenswürdigen Quellen stammen. Die Angabe `%S` am Beginn des Format Strings veranlasst `aptitude`, ein *U* zu produzieren, sollte der entsprechende Fall eintreten. Der angeflanschte Aufruf von `egrep` benutzt einen regulären Ausdruck, um für diejenigen Zeilen einen Treffer zu landen, die mit einem großen *U* beginnen. Im Beispiel ist es das ausgedachte Paket `libblafaselsonstwas`.

**Pakete aus nicht-vertrauenswürdiger Quelle herausfischen**

```bash
$ aptitude search '~i' -F '%S %p' | egrep '^U
U libblafaselsonstwas
$
```

### 10.9 Für `aptitude` die Ausgabebreite festlegen


**Begrenzung der Ausgabe auf eine feste Breite**

```bash
$ aptitude search -w 40 debtags
i  debtags   - Aktivierte die Unterteilung
p  debtags:i386  - Aktivierte die Unterteilung
p  python-debt - Vergleicht >hardware:
 p python3-debt - Vergleicht >hardware:
$
```

Dieser Wert korrespondiert mit der Einstellung `Aptitude::CmdLine::Package-Display-Width` in der Konfigurationsdatei zu `aptitude`. 

Debian-Paketmanagement
10.10 Bei aptitude die Ausgabe sortieren

Möchten Sie die Ausgabe darüberhinaus noch sortieren, hilft Ihnen der Schalter --sort (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.


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>
```

10.11 aptitude-Gruppierung

10.11.1 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.

10.11.2 Textoberfläche

Auch die textbasierte Bedienoberfläche ist nicht in Stein gemeisselt und lässt sich anpassen. Dazu gehören die Anordnung und Breite der Spalten zu den Paketen und deren Informationen sowie die Aufteilung auf dem Bildschirm — horizontal oder vertikal geteilt.
10.11.2.1 Anordnung und Breite der Spalten

Die Darstellung entnimmt Aptitude der Konfigurationsgruppe `Aptitude::UI::Default-Package-View`. Diese beinhaltet eine Liste der Komponenten, die in der TUI auftauchen. Jede Komponente bennennet Sie und konfigurieren Sie über mindestens die nachfolgenden vier Angaben genauer:

**Beschreibung für eine Komponente der Aptitude-TUI**

```plaintext
Komponentenname Typ {
  Row Zeile;
  Column Spalte;
  Width Breite;
  Height Höhe;
};
```

Der **Komponentenname** ist frei wählbar, darf aber nur aus einem einzigen Wort ohne Trennzeichen bestehen. Der **Typ** bestimmt, was in der Komponente dargestellt wird. Zur Auswahl stehen:

- **Description**
  Informationsfeld, bspw. für die Beschreibung des aktuell ausgewählten Debianpaketes

- **MainWidget**
  Feld mit der Liste der Pakete; dieses muss genau einmal vorkommen

- **Static**
  Feld mit statischem Text

Die Angaben für Zeile, Spalte, Breite und Höhe entsprechen denen einer Tabelle, wobei die obere, linke Ecke die Position 0,0 hat. Mit den nachfolgenden Zusatzoptionen gestalten Sie die Darstellung einer Spalte und deren jeweiligen Inhalt noch genauer:

- **ColAlign** Ausrichtung
- **ColExpand**
- **Color**
- **ColShrink**
- **Column**
- **Columns**
- **ColumnsCfg**
- **Height**
- **PopUpDownKey**
- **PopUpDownLinked**
- **Row**
- **RowAlign**
- **RowExpand**
- **RowShrink**
- **Visible**
- **Width**

**Beispiel für eine Komponente**

Todo
10.11.2.2 Aufteilung auf dem Bildschirm

Dieses Thema besprechen wir im Abschnitt 10.12 genauer.

10.12 aptitude-Farbschema anpassen

10.12.1 Zwischen aptitude-Themes wechseln

Das Aussehen von der aptitude-Bedienoberfläche hängt von drei Einstellungen ab — den gewählten Farben, der Anordnung der Ausgabeelemente für die TUI und dem Wert der Konfigurationsoption aptitude::Theme.


Bleibt noch obige Konfigurationsoption aptitude::Theme, deren Bedeutung es zu klären gilt. Für diese wählen Sie den Namen des von Ihnen gewünschten Themes aus. aptitude liefert die folgenden zwei Themes mit:

Dselect
das aptitude-Standardtheme, welches an die Bedienoberfläche des Werkzeugs dselect angelehnt ist. Oben befindet sich der Paketbaum aus Kategorien und den sich darin befindlichen Softwarepaketen, darunter die Paketbeschreibung zum jeweils ausgewählten Paket

Vertical-Split
Anordnung, in der die Ausgabe vertikal geteilt ist. In der linken Spalte befindet der Paketbaum aus Kategorien und den sich darin befindlichen Softwarepaketen, in der rechten Spalte die Paketbeschreibung zum jeweils ausgewählten Paket

10.12.2 Standardvorgaben

aptitude kommt mit einer Auswahl an Standardfarben. Diese sind in Tabelle 6.2 in Abschnitt 6.3.2 benannt.

10.12.3 Eigene Farben vergeben

Gefällt Ihnen die mitgelieferte Farbauswahl nicht, bauen Sie sich am besten ihr eigenes Theme. Dazu legen Sie für die einzelnen Strukturelemente eigene Farben fest (siehe [aptitude-dokumentation-text-colors-and-styles]). Anpassbare Elemente sind:

- Bullet
- ChangelogNewerVersion
- Default
- DepBroken
- DisabledMenuEntry
- DownloadHit
- DownloadProgress
- EditLine
- Error
- Header
- HighlightedMenuBar
Als Farben steht Ihnen eine Grundauswahl aus 8 Farben zur Verfügung. Tabelle 10.2 stellt diese zusammen.
Tabelle 10.2: Farbauswahl bei aptitude

<table>
<thead>
<tr>
<th>Farbname</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>black</td>
<td>schwarz</td>
</tr>
<tr>
<td>blue</td>
<td>blau</td>
</tr>
<tr>
<td>cyan</td>
<td>hellblau</td>
</tr>
<tr>
<td>green</td>
<td>grün</td>
</tr>
<tr>
<td>magenta</td>
<td>dunkelrosa</td>
</tr>
<tr>
<td>red</td>
<td>rot</td>
</tr>
<tr>
<td>white</td>
<td>weiss</td>
</tr>
<tr>
<td>yellow</td>
<td>gelb</td>
</tr>
</tbody>
</table>

Ergänzend zur Farbauswahl erlaubt aptitude zusätzliche Hervorhebungen. Tabelle 10.3 fasst die Einstellungen dafür zusammen

Tabelle 10.3: Hervorhebungen und deren Bedeutung bei aptitude

<table>
<thead>
<tr>
<th>Hervorhebung</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>blink</td>
<td>Enables blinking text.</td>
</tr>
<tr>
<td>bold</td>
<td>Makes the foreground color of the text (or the background if reverse video is enabled) brighter.</td>
</tr>
<tr>
<td>dim</td>
<td>May cause text to be extra-dim on some terminals. No effect has been observed on common Linux terminals.</td>
</tr>
<tr>
<td>reverse</td>
<td>Swaps the foreground and background colors. Many visual elements flip this attribute to perform common highlighting tasks.</td>
</tr>
<tr>
<td>standout</td>
<td>This enables “the best highlighting mode of the terminal”. In xterms it is similar, but not identical to, reverse video; behavior on other terminals may vary.</td>
</tr>
<tr>
<td>underline</td>
<td>Enables underlined text.</td>
</tr>
</tbody>
</table>

• gibt es vorgefertigte Themes zum Download?

• 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


`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


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

```bash
# 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
```
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.

<table>
<thead>
<tr>
<th>Taste</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Paket installieren oder aktualisieren (install oder upgrade)</td>
</tr>
<tr>
<td>-</td>
<td>Paket entfernen (remove)</td>
</tr>
<tr>
<td>_</td>
<td>Paket vollständig entfernen (purge)</td>
</tr>
<tr>
<td>:</td>
<td>Paketversion behalten (keep)</td>
</tr>
<tr>
<td>=</td>
<td>Paketversion dauerhaft beibehalten (hold)</td>
</tr>
<tr>
<td>L</td>
<td>Paket nochmals installieren (reinstall)</td>
</tr>
<tr>
<td>U</td>
<td>alle aktualisierbaren Pakete zur Aktualisierung vormerken</td>
</tr>
</tbody>
</table>

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 8.38)

upgrade
listet alle Pakete auf, die durch eine neuere Version ersetzt werden (siehe Abschnitt 8.41)

downgrade
listet alle Pakete auf, die durch eine ältere Version ersetzt werden (siehe Abschnitt 8.42)
**remove**
listet alle Pakete auf, die gelöscht werden (siehe Abschnitt 8.43)

**purge**
listet alle Pakete auf, die vollständig gelöscht werden (siehe Abschnitt 8.43)

**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**

```bash
# aptitude search '~ainstall'
pi cssed - graphical CSS editor
# aptitude search '~aremove'
id apt-doc - Dokumentation für APT
# aptitude search '~auupgrade'
iu iceweasel - Webbrowser auf Basis von Firefox
iu libmozjs24d - Mozilla SpiderMonkey JavaScript library
iu xulrunner-24.0 - XUL + XPCOM application runner
#
```


**Pakete darstellen, die nicht in dem bestehenden Zustand gehalten werden**

```bash
# 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 8.38),
- Pakete, die zurückgehalten werden (siehe Abschnitt 2.15),
- Pakete, die entfernt werden (siehe Abschnitt 8.43) und
- Pakete, die aktualisiert werden (siehe Abschnitt 8.41).
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 vorgemerk, 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 verschiedenen Farben lesen Sie in Abschnitt 6.3.2 und Abschnitt 10.12.

![Abbildung 11.1: Paketoperationen anzeigen, die zur Ausführung anstehen](image)

### 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

```bash
# aptitude install -s
Die folgenden NEUEN Pakete werden zusätzlich installiert:
cssed
Die folgenden Pakete werden ENTFERNT:
atp-doc
Die folgenden Pakete werden aktualisiert:
iceweasel libmozjs24d xulrunner-24.0
3 Pakete aktualisiert, 1 zusätzlich installiert, 1 werden entfernt und 19 nicht aktualisiert.
Möchten Sie fortsetzen? [Y/n/?]
Pakete würden heruntergeladen/installiert/entfernt werden.
#```

### Automatisches Ausführen

Kombinieren Sie obigen `aptitude`-Aufruf zur Simulation mit dem Parameter `--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 Unterkomando 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).

![](image)

**Abbildung 11.2: Vormerkungen abbrechen**

### 11.6 Vormerkungen ausführen


```
# 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.
```
Debian-Paketmanagement

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


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.

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 verschieden Programm 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 zueinander passen. 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 8.15), die Paketabhängigkeiten anzeigen (siehe Abschnitt 8.19), die Herkunft der Pakete klären (siehe Abschnitt 8.14) und über den Paketinhalt suchen (siehe dazu Abschnitt 8.24 und Abschnitt 8.26).

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 Seitenwechsel 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 unvorhersehbaren Ergebnis.


Im Alltag hat sich beispielsweise 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


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 Ihnen auch eine Hinweis 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.

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


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.


**Klassifikation der Benutzerschnittstelle über die Facette interface (Ausschnitt)**

<table>
<thead>
<tr>
<th>Facet: interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: User Interface</td>
</tr>
<tr>
<td>What kind of user interface the package provides</td>
</tr>
<tr>
<td>Status: needing-review</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag: interface::3d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: Three-Dimensional</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag: interface::commandline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: Command Line</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag: interface::text-mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: Text-based Interactive</td>
</tr>
</tbody>
</table>


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

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 Schlagwörtern. Weiterhin gehört auch eine statistische Auswertung dazu, um die Menge und die Verteilung der genutzten Schlagworte nachvollziehen zu können [Debian-Debtags-Statistics].


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


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).


**Recherche mittels ara nach optionalen Paketen zum Window Manager XFCE (Auswahl)**

```bash
$ ara -fields Package,Size,Maintainer:30 -table 'section=utils & (depends:(xfce) | size ←<10000) & priority=optional'

<table>
<thead>
<tr>
<th>Package</th>
<th>Size</th>
<th>Maintainer</th>
</tr>
</thead>
<tbody>
<tr>
<td>acpitail</td>
<td>8340</td>
<td>Debian Acpi Team &lt;pkg-acpi-...</td>
</tr>
<tr>
<td>athena-jot</td>
<td>9876</td>
<td>Francesco Paolo Lovergine &lt;...</td>
</tr>
<tr>
<td>autotrash</td>
<td>9796</td>
<td>Lorenzo De Liso &lt;blackz@ubu...</td>
</tr>
<tr>
<td>binclock</td>
<td>9540</td>
<td>Nico Golde <a href="mailto:nion@debian.org">nion@debian.org</a></td>
</tr>
<tr>
<td>colortest-python</td>
<td>9052</td>
<td>Jari Aalto &lt;jari.aalto@cant...</td>
</tr>
<tr>
<td>createfp</td>
<td>9982</td>
<td>Rene Engelhard &lt;rene@debian...</td>
</tr>
<tr>
<td>ddir</td>
<td>9776</td>
<td>Jari Aalto &lt;jari.aalto@cant...</td>
</tr>
<tr>
<td>eatmydata</td>
<td>7778</td>
<td>Modestas Vainius &lt;modax@deb...</td>
</tr>
<tr>
<td>ksshaskpass</td>
<td>9426</td>
<td>Armin Berres &lt;armin+debian8...</td>
</tr>
<tr>
<td>laptop-detect</td>
<td>5212</td>
<td>Otavio Salvador &lt;otavio@deb...</td>
</tr>
</tbody>
</table>
```

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

Abbildung 13.2: Debian Tags zum Paket xara-gtk (zuletzt in Debian 9 Stretch)
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**

```bash
$ 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**

```bash
$ 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**

```bash
$ 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`), 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++`).


**Auflistung der vergebenen Schlagwörter ohne Wert für das Paket apt-move**

```bash
$ 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](image)


13.5.3 Graphische Programme


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.

13.5.4 Über den Webbrowser


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

13.6.1.1 Suche mittels debtags

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 Deftags-Eintrag. Dieser Eintrag besteht wiederum aus drei Teilen – einem Schlag-
wort (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 Beispielaufbau sehen Sie ei-
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.

```bash
$ 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.

```bash
$ 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 Suchbe-
griff *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 eingesortiert
wurden. Daher umfasst die Recherche die beiden Tags `interface::x11` und `game::simulation`.

```bash
$ 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+
libopencexeneograph-dev - 3D scene graph, development files
libopencexeneograph80 - 3D scene graph, shared libs
lincity-ng - City simulator game with polished graphics
oolite - space sim game, inspired by Elite
openccity - 3D city simulator game
opensas - 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
```
An dieser Stelle hilft Ihnen auch das Paket `dctrl-tools` weiter – jetzt jedoch mit dem Programm `grep-debtags`. Mit dem nachfolgendem Aufruf erhalten Sie eine Liste aller verfügbaren Pakete zu leichtgewichtigen Web-

browsern, 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 einzige 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
```

13.6.1.2 Suche mit `axi-cache`


Abbildung 13.4: Ergebnis der Paketsuche nach dem Tag `use::searching` mit Hilfe von `axi-cache`
13.6.2 Textoberfläche von `aptitude`


![Abbildung 13.5: Auswahl des Debtags-Browsers in `aptitude`](image)

13.6.3 Graphische Programme

In dieser Kategorie bleiben aus der Liste der Werkzeuge zur Paketverwaltung nur PackageSearch (siehe Abbildung 13.6) übrig. Bei PackageSearch stöbern Sie über die Liste oben rechts und selektieren daraus die gewünschten Einträge.
13.6.4 Suche über den Webbrowser


Abbildung 13.7: Suche anhand der Debtags über den Webbrowser

Abbildung 13.8: 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 13.9 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 13.9: 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 Schlagwörter 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.


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::nttp - 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**

```bash
$ 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: https://en.wikipedia.org/wiki/Post_Office_Protocol
Link: http://www.ietf.org/rfc/rfc1939.adoc
```

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 13.10).

![Abbildung 13.10: Informationen zur Facette protocol::pop3](image)

### 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 https://debtags.debian.org/
Kapitel 14

Mehrere Pakete in einem Schritt ändern

14.1 Mit apt-get


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

```bash
# 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).

<table>
<thead>
<tr>
<th>Taste</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>+paket</td>
<td>Paket installieren (install)</td>
</tr>
<tr>
<td>-paket</td>
<td>Paket entfernen (remove)</td>
</tr>
<tr>
<td>_paket</td>
<td>Paket vollständig inklusive Konfigurationsdateien entfernen (purge)</td>
</tr>
<tr>
<td>:paket</td>
<td>Paketversion behalten (keep)</td>
</tr>
<tr>
<td>=paket</td>
<td>Paketversion dauerhaft beibehalten (hold)</td>
</tr>
</tbody>
</table>

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 gibt
  - Änderungen am [Y/n/?] Prompt
Kapitel 15

Ausgewählte Pakete aktualisieren


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 für diesen Schritt und die Wartung etwas mehr Zeit 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.

15.1.1 Auf der Kommandozeile

Hier 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.


Beispiel für Einzelaktualisierung mit aptitude safe-upgrade

ToDo: Beispiel
15.1.2 Über die Textoberfläche von aptitude

- Zweig/Kategorie Sicherheitsaktualisierungen
  - Paket markieren mit `-`
  - Paketvorschau mit `g`
  - Paketaktualisierung mit `g`

- ToDo: da muss aber noch mehr sein . . . .

15.1.3 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**

```bash
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


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`**

```bash
# apt-mark hold wireshark
wireshark auf Halten gesetzt.
```

**Entfernen der Markierung `hold` für das Paket `wireshark` mittels apt-mark**

```bash
# apt-mark unhold wireshark
Halten-Markierung für wireshark entfernt.
```


**Auflistung aller Pakete, die gehalten werden**

```bash
# apt-mark showhold
wireshark
```
16.2 Textoberflächen

- `aptitude`

16.3 Graphische Programme

- `Synaptic`
  - Menüpunkt: Version sperren (siehe Abbildung 16.1)
    * farbige Hervorhebung (default: rot) in der Auswahlliste der Pakete
    * Symbol eines Hängeschloß über dem Auswahlkästchen
    * Kontextmenü mit den Paketaktionen (erreichbar über die rechte Maustaste) nicht mehr aufrufbar, Aktionen nur noch über das Menü
  - geht nur bei bereits installierten Paketen

Abbildung 16.1: Setzen des Paketflags `hold` in Synaptic
Kapitel 17

Fehlende Pakete bei Bedarf hinzufügen

• bisher:
  – Softwareauswahl/Pakete ist fest
  – Liste der installierte Pakete ist statisch, d.h. ändert sich nicht im laufenden Betrieb
• Bedarf ändert sich:
  – neue Hardware kommt hinzu
  – Software fehlt bzw. wird benötigt
• nachfolgend vorgestellte Pakete zeigen, in welchem Rahmen eine Reaktion auf diese Bedarfsänderung möglich ist

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.


Hinter Isenkram verbirgt sich eine Art Benachrichtigungsdienst, der überprüft, ob die benötigten Pakete für die neue Hardware bereits auf ihrem 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

17.2.1 Empfehlungen mittels command-not-found

- Wer nicht gleich das Paket installiert haben will, sondern nur beim Aufruf auf der Kommandozeile den Hinweis bekommen, welches Paket installiert werden müsste, für den ist vermutlich das Paket `command-not-found` das richtige.
Kapitel 18

Alternative Standard-Programme mit Debi ans Alternativen- System


Ähnliches leisten die Programme select-editor, sensible-browser, sensible-editor und sensible-pager aus dem 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.


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 alternatives aus dem gleichnamigen Paket bereit. In Abbildung 18.1 sehen Sie exemplarisch die Möglichkeiten für die Gruppe x-www-browser.

1X Desktop Group (XDG), seit dem Jahr 2000 freedesktop.org
18.1 Hintergrund: Warum alternative Standardprogramme?


Das betrifft auf der einen Seite sehr generische Funktionalitäten wie z.B. die eines Web-browsers, eines (Text-)Editors oder eines Pagers, aber auch sehr spezifische Funktionalitäten, für die einfach mehrere Implementierungen vorliegen. Zu Letzteren gehören bspw. das Schweizer Taschenmesser für TCP/IP namens netcat oder die Skriptsprache awk. Letztere liegt in drei Varianten vor — in der eher umfangreicherem Implementation des GNU Projektes (gawk), in der kleineren und schnelleren Implementation von Mike Brennan (mawk) und in der originalen Awk-Implementation [awk].


In Debian 9 Stretch ist dagegen GCC wieder nur noch in einer Version enthalten, nämlich 6.3.0. Dafür ist OpenSSL hingegen in zwei verschiedenen Versionen enthalten — 1.0.0 und 1.0.2. Hintergrund ist hier, dass bestimmte Programme (noch) nicht sauber mit der neueren OpenSSL-Version kompiliert werden können und die dazu notwendigen Änderungen nicht-trivial sind. So ist z.B. Apache gegen OpenSSL 1.0.2 gelinkt, weil noch nicht alle SSL-nutzenden Apache-Module sauber mit OpenSSL 1.1 zusammenarbeiten. Eine Standard-Version in dem Sinne, wie es sie beim GCC gibt, existiert hier jedoch nicht. Gegen welche OpenSSL-Version ein Programm kompiliert und später gelinkt wird, hängt davon ab, welches der beiden sich gegenseitig ausschließenden Pakete libssl-dev(OpenSSL 1.1) oder libssl1.0-dev(OpenSSL 1.0.2) installiert ist.

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:

- ganz ohne graphische Benutzeroberfläche (emacs23-nox bzw. emacs24-nox).

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.

Beispiel-Ausgabe von Axel's Thinkpad und mit einer durchaus nicht ganz üblichen Auswahl von `update-alternatives --get-selections` (massiv gekürzt)

```bash
$ 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
c              auto    /usr/bin/gcc
cpp            auto    /usr/bin/cpp
csh            auto    /bin/bsh-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
director        manual  /usr/bin/zile
dmtes           auto    /usr/bin/emacs24-x
dmtesclient     auto    /usr/bin/emacsclient.emacs24
dmtesекс        auto    /usr/bin/nex
gnome-text-editor auto    /usr/bin/leafpad
gnome-www-browser auto    /usr/bin/opera
text2markdown   auto    /usr/bin/text2markdown.py2
textbrowser      auto    /usr/bin/info
tar             auto    /usr/bin/fsttar
java            auto    /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java
ksh             auto    /bin/ksh93
locate          auto    /usr/bin/locate
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            auto    /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/nvview
wesnoth         auto    /usr/games/wesnoth-1.10
www-browser     auto    /usr/bin/links2
x-cursor-theme   manual  /etc/X11/curxors/white.ceramic
x-session-manager auto    /usr/bin/choosewm
```
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
```


**Ausgabe der ausgewählten und verfügbaren Alternativen für Awk**

```
$ update-alternatives --display awk
awk - automatischer Modus
/usr/bin/gawk - Priorität 10
Slave awk.1.gz: /usr/share/man/man1/gawk.1.gz
Slave nawk: /usr/bin/gawk
Slave awk.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 [RFC822] 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.
```

---

Hinweise über die automatische Auswahl von Alternativen bei der Paketinstallation

Listing

```
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/edit (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

[...]
```


Ändern des systemweiten Standardeditors von einer automatischen Wahl auf zile

Listing

```
$ update-alternatives --config editor
Es gibt 10 Auswahlmöglichkeiten für die Alternative editor (welche /usr/bin/editor bereitstellen).

<table>
<thead>
<tr>
<th>Auswahl</th>
<th>Pfad</th>
<th>Priorität</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/usr/bin/vim.gtk</td>
<td>50</td>
<td>automatischer Modus</td>
</tr>
<tr>
<td>1</td>
<td>/bin/ed</td>
<td>-100</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>2</td>
<td>/bin/elvis-tiny</td>
<td>10</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>3</td>
<td>/bin/nano</td>
<td>40</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>4</td>
<td>/usr/bin/emacs24</td>
<td>0</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>5</td>
<td>/usr/bin/medit</td>
<td>25</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>6</td>
<td>/usr/bin/nvi</td>
<td>19</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>7</td>
<td>/usr/bin/vigor</td>
<td>-150</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>8</td>
<td>/usr/bin/vim.gtk</td>
<td>50</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>9</td>
<td>/usr/bin/vim.nox</td>
<td>40</td>
<td>manueller Modus</td>
</tr>
<tr>
<td>10</td>
<td>/usr/bin/zile</td>
<td>30</td>
<td>manueller Modus</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Selection</th>
<th>Path</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/usr/lib/aspell/de-neu.multi</td>
<td>1996</td>
<td>auto mode</td>
</tr>
<tr>
<td>* 1</td>
<td>/usr/lib/aspell/de-alt.multi</td>
<td>1901</td>
<td>manual mode</td>
</tr>
<tr>
<td>2</td>
<td>/usr/lib/aspell/de-neu.multi</td>
<td>1996</td>
<td>manual mode</td>
</tr>
</tbody>
</table>
Kapitel 19

Backports


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 der Veröffentlichung stable sind Sicherheitsupdates oder die Behebung größerer 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 in das Verzeichnis /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


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


19.3 Debian Backports


19.4 Welche Pakete gibt es als offiziellen Backport?


19.5 Welche Versionen gibt es als offizielle Backports?


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 9 Stretch und Debian 8 Jessie mit den dem Distributionsbereich main sieht der jeweilige Eintrag wie folgt aus:

Eintrag zu Debian Backports für Debian 9 Stretch

```
# Backports
deb http://ftp.debian.org/debian/stretch-backports main
```

Eintrag zu Debian Backports für Debian 8 Jessie

```
# Backports
deb http://ftp.debian.org/debian/jessie-backports main
```

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
```


Das Paket asciidoc steht bspw. nicht für Debian 7 Wheezy bereit, existiert jedoch in den Backports. Um dieses aus den Backports 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 Die installierten Pakete anzeigen

Wie bereits genannt, gliedern sich die Pakete aus Debian Backports recht nahtlos in den Paketbestand ein. Möchten Sie herausfinden, welche der installierten Pakete auf ihrem System aus den Backports stammen, kommt ihnen das Namens- und Versionsschema von Debian entgegen (siehe Benennung einer Paketdatei in Abschnitt 2.11).

In der Versionsangabe des Pakets ist die Zeichenkette ~bpo plus die Versionsnummer der Debianveröffentlichung enthalten, so bspw. ~bpo8 für eine Rückportierung auf Debian 8 Jessie und ~bpo9 für eine Rückportierung auf Debian 9 Stretch. Diese Angabe ist soweit verlässlich, da sich die meisten Entwickler brav an diese Konvention für die Benennung halten. Zusätzlich achten die FTP-Master für Debian Backports recht pedantisch darauf, dass diese Schreibweise eingehalten wird.

Mit einem beherzten Griff zu dpkg, grep und awk filtern Sie aus der Paketliste alle Pakete samt Versionsnummer heraus, die diese Zeichenkette enthalten:

Auflisten der installierten Pakete samt Versionsnummer aus Debian Backports

```
# dpkg -l | grep ~bpo | awk...
```
$ dpkg -l | grep -bpo8 | awk '{print $2 " " $3}'
libroutino-slim0 3.1.1-1~bpo8+1
libroutino0 3.1.1-1~bpo8+1
qmapshack 1.7.2-1~bpo8+1
routino 3.1.1-1~bpo8+1
routino-common 3.1.1-1~bpo8+1
$

Geht es Ihnen nur um die Paketnamen, vereinfacht sich der Aufruf wie folgt — Sie filtern mittels awk und einem Regulären Ausdruck und geben danach nur noch die zweite Spalte jeder Zeile aus:

**Auflisten der installierten Pakete aus Debian Backports**

$ dpkg -l | awk '/~bpo8/{print $2}'
libroutino-slim0
libroutino0
qmapshack
routino
routino-common
$


Die Variante a) mit aptitude kombiniert mehrere Aufrufparameter — so search -i zur Suche in den installierten Paketen und dazu ?narrow() zur weiteren Eingrenzung der Auswahl. ~VCURRENT bezieht sich dabei auf die aktuelle Debianversion, ~Abports auf die Pakete, die aus den Paketquellen (Archiv) kommen, in denen Backports im Archivnamen vorkommt.

**Auflisten der installierten Pakete aus Debian Backports mittels aptitude**

$ aptitude search '~i ?narrow(~VCURRENT, ~Abports)'
i A libroutino-slim0 - Routino slim routing library
i A libroutino0 - Routino routing library
i qmapshack - GPS mapping (GeoTiff and vector) and GPSr
i A routino - Set of tools to find a path between two po
i A routino-common - Routino data
$


Die Variante b) auf der Basis von apt und fgrep liefert das gleiche Ergebnis, jedoch eine andere Ausgabe. Jede Zeile beinhaltet den Paketnamen samt Repository, aus dem Paket stammt, sowie die Versionsnummer, die Architektur und den Installationsstatus.

**Auflisten der installierten Pakete aus Debian Backports mittels apt**

$ apt list --installed | fgrep backports
WARNING: apt does not have a stable CLI interface yet. Use with caution in scripts.
libroutino-slim0/jessie-backports,now 3.1.1-1~bpo8+1 amd64 [Installiert, automatisch]
libroutino0/jessie-backports,now 3.1.1-1~bpo8+1 amd64 [Installiert, automatisch]
qmapshack/jessie-backports,now 3.1.1-1~bpo8+1 amd64 [automatischer Installiert]
routino/jessie-backports,now 1.7.2-1~bpo8+1 amd64 [installiert]
routino-common/jessie-backports,now 3.1.1-1~bpo8+1 all [installiert, automatisch]
$

Debian-Paketmanagement
19.8 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].

19.9 Backports bei Ubuntu


19.10 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 meinst 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.


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


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


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

<table>
<thead>
<tr>
<th>Priorität</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>unter 0</td>
<td>Das Paket wird niemals installiert.</td>
</tr>
<tr>
<td>zwischen 0 und 100</td>
<td>Das Paket wird nur dann installiert, wenn noch keine Version davon auf dem System installiert ist.</td>
</tr>
<tr>
<td>zwischen 100 und 500</td>
<td>Das Paket wird installiert, es sei denn, es gibt eine Version, die zu einer anderen Veröffentlichung gehört, oder die bereits vorhandene Version ist neuer.</td>
</tr>
<tr>
<td>zwischen 500 und 990</td>
<td>Das Paket wird installiert, es sei denn, es gibt eine Version, die zur festgelegten Zielveröffentlichung gehört, oder die bereits vorhandene Version ist neuer.</td>
</tr>
<tr>
<td>zwischen 991 und 1000</td>
<td>Das Paket wird immer installiert, es sei denn, die bereits vorhandene Version ist neuer.</td>
</tr>
<tr>
<td>Wert größer als 1000</td>
<td>Das Paket wird immer installiert, auch wenn das ein Downgrade auf eine ältere Version bedeutet.</td>
</tr>
</tbody>
</table>

20.4 Paketweise festlegen

Für diesen Fall besteht eine Liste mit Einträgen für einzelne Pakete und ganze Paketgruppen. Bis Debian 6 *Squeeze* war die Datei `/etc/apt/preferences` die einzige Stelle, an der Prioritäten für Paketnamen, Veröffentlichungen, Hersteller oder Versionen eingetragen werden konnten.

Ab Debian 7 *Wheezy* wurde Unterstützung für ein Verzeichnis `/etc/apt/preferences.d/` eingeführt. 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**

```bash
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**

**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

<table>
<thead>
<tr>
<th>Parameter und Schlüsselwort</th>
<th>Bedeutung</th>
<th>Beispiel</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (archive)</td>
<td>Veröffentlichung (siehe Abschnitt 2.10)</td>
<td>unstable</td>
</tr>
<tr>
<td>c (component)</td>
<td>Distributionsbereich (siehe Abschnitt 2.9)</td>
<td>main</td>
</tr>
<tr>
<td>l (label)</td>
<td>Bezeichner</td>
<td>Debian</td>
</tr>
<tr>
<td>n (name)</td>
<td>Aliasnamen der Veröffentlichung (siehe Abschnitt 2.10.2)</td>
<td>Stretch</td>
</tr>
<tr>
<td>o (origin)</td>
<td>Herkunft</td>
<td>Debian</td>
</tr>
<tr>
<td>v (version)</td>
<td>explizite Versionsnummer (siehe Abschnitt 2.11)</td>
<td>6.0.3</td>
</tr>
</tbody>
</table>

## 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**

<table>
<thead>
<tr>
<th>Package: *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin: release v=7.5, l=Debian</td>
</tr>
<tr>
<td>Pin-Priority: 1000</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Package: samba*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin: v=3.5.6*</td>
</tr>
<tr>
<td>Pin-Priority: 1000</td>
</tr>
</tbody>
</table>

**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 bestimmen**

<table>
<thead>
<tr>
<th>Package: *gnome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin: origin ftp.informatik.tu-berlin.de</td>
</tr>
<tr>
<td>Pin-Priority: 600</td>
</tr>
</tbody>
</table>
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]).


21.3 Software selbst übersetzen und als deb-Paket einspielen


---

Hinweis auf dh-make-perl

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 Blogbeitrag „Building Debian packages of Perl modules“ [Kemp-dh-make-perl].

<table>
<thead>
<tr>
<th>Paketformat</th>
<th>Schalter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian-Paket (deb)</td>
<td>-D</td>
</tr>
<tr>
<td></td>
<td>-t debian</td>
</tr>
<tr>
<td></td>
<td>--type=debian</td>
</tr>
<tr>
<td>RPM-Paket (rpm)</td>
<td>-R</td>
</tr>
<tr>
<td></td>
<td>-t rpm</td>
</tr>
<tr>
<td></td>
<td>--type=rpm</td>
</tr>
<tr>
<td>Slackware-Paket (tgz)</td>
<td>-S</td>
</tr>
<tr>
<td></td>
<td>-t slackware</td>
</tr>
<tr>
<td></td>
<td>--type=slackware</td>
</tr>
</tbody>
</table>


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


checkinstall --help
Tabelle 21.2: Spezifische Schalter für ein Debian-Binärpaket

<table>
<thead>
<tr>
<th>Schalter</th>
<th>Bedeutung</th>
</tr>
</thead>
<tbody>
<tr>
<td>--pkgname=name</td>
<td>Name des Pakets</td>
</tr>
<tr>
<td>--pkgversion=version</td>
<td>Versionsnummer des Pakets</td>
</tr>
<tr>
<td>--arch Architektur</td>
<td>Architektur des Pakets</td>
</tr>
<tr>
<td>--pkgarch=Architektur</td>
<td>Architektur des Pakets</td>
</tr>
<tr>
<td>--pkgrelease=Release</td>
<td>Angabe der Veröffentlichung</td>
</tr>
<tr>
<td>--pkglicense=Lizenz</td>
<td>Angabe der Lizenz zum Paket</td>
</tr>
<tr>
<td>--pkggroup=Gruppe</td>
<td>Benennung der Paketkategorie</td>
</tr>
<tr>
<td>--pkgsource=Quelle</td>
<td>Angabe der Quelle zum Paket</td>
</tr>
<tr>
<td>--pkgaltsource=Quelle</td>
<td>alternative Angabe der Quelle zum Paket</td>
</tr>
<tr>
<td>--pakdir=Verzeichnis</td>
<td>Zielverzeichnis, in dem das Paket gespeichert wird</td>
</tr>
<tr>
<td>--maintainer=Emailadresse</td>
<td>Emailadresse des Paketmaintainers</td>
</tr>
<tr>
<td>--provides=Liste</td>
<td>Name der Pakete, die es bereitstellt</td>
</tr>
<tr>
<td>--requires=Liste</td>
<td>Name der Pakete, die das Paket benötigt</td>
</tr>
<tr>
<td>--conflicts=Liste</td>
<td>andere Pakete, mit denen das Paket in Konflikt steht</td>
</tr>
<tr>
<td>--replaces=Liste</td>
<td>andere Pakete, die dieses Paket ersetzt</td>
</tr>
<tr>
<td>--dpkgflags=Flags</td>
<td>Flags, die an dpkg zur Installation mitgegeben werden</td>
</tr>
<tr>
<td>--nodoc</td>
<td>keine Dokumentation in das Paket einfügen</td>
</tr>
</tbody>
</table>

21.4 Beispiel

ToDo

21.5 Vor- und Nachteile


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.


21.5.1 Weitere noch unbearbeitete Notizen

• Vorteile:
  – installiert automatisch die zusätzlichen, bisher noch nicht installierten Header-Files nach
  * stimmt das?
Kapitel 22

Metapakete bauen

Metapakete sind spezielle Binärpakete, die keine Daten, sondern lediglich Abhängigkeiten auf andere Pakete beinhalten. Genauer erklären wir Ihnen diese Paketvariante unter „Übergangspakete, Metapakete und Tasks“ in Abschnitt 2.7.2.


22.1 Vorbereitungen

Zur Erstellung des Metapakets meta-mc greifen wir auf das Debianpaket equivs [Debian-Paket-equivs] zurück, was diesen Vorgang für alle Seiten sehr stark vereinfacht. Das Paket equivs bringt die beiden Werkzeuge equivs-control und equivs-build mit.

Ausgangspunkt ist das Erzeugen einer Beschreibung des zu erstellenden Metapakets. Dazu benutzen Sie equivs-control. Dieses erwartet einen Dateinamen, in der die Beschreibung landet — nachfolgend genannt ns-control:

Beschreibungsdatei für das Metapaket erzeugen

```
$ equivs-control ns-control
```

Die mittels equivs-control erzeugte Beschreibung ist nur eine Vorlage, die Sie nun noch entsprechend ändern, damit es auch auf das Metapaket meta-mc passt. Das betrifft insbesondere die Felder Section (Paketkategorie), Priority (Priorität des Pakets), Package (Paketname des Metapakets), Maintainer (Name des Paketbetreuers), Depends (Paket hängt ab von) und Description (Paketbeschreibung). Passende Felder kommentieren Sie aus und hinterlegen den gewünschten Wert. In unserem Fall sieht das wie folgt aus:

Angepasste Beschreibungsdatei ns-control für das Metapaket meta-mc

```
% Section: Filesystem
% Priority: optional
% Package: meta-mc
% Maintainer: Debian-Paketmetapaket
% Depends: mc [mc-data]
% Description: Metapaket für den Midnight Commander
```

```
```

```
```
### Commented entries have reasonable defaults.
### Uncomment to edit them.
# Source: <source package name; defaults to package name>
Section: misc
Priority: optional
# Homepage: <enter URL here; no default>
Standards-Version: 3.9.2
Package: meta-mc
# Version: <enter version here; defaults to 1.0>
Maintainer: Frank Hofmann <frank.hofmann@efho.de>
# Pre-Depends: <comma-separated list of packages>
Depends: mc, mc-data
# Recommends: <comma-separated list of packages>
# Provides: <comma-separated list of packages>
# Replaces: <comma-separated list of packages>
# Architecture: all
# Multi-Arch: <one of: foreign|same|allowed>
# Copyright: <copyright file; defaults to GPL2>
# Changelog: <changelog file; defaults to a generic changelog>
# Readme: <README.Debian file; defaults to a generic one>
# Extra-Files: <comma-separated list of additional files for the doc directory>
# Files: <pair of space-separated paths; First is file to include, second is destination>
# <more pairs, if there's more than one file to include. Notice the starting space>
Description: Installs the Midnight Commander
Installs the Midnight Commander

---

### Anmerkung
Beachten Sie bitte die Leerzeile am Ende der Beschreibungsdatei ns-control. Ohne diese kann das Paket später nicht sauber gebaut werden.

#### 22.2 Das Paket bauen


**Bauen des Metapakets auf der Basis der Beschreibungsdatei**

```
$ equivs-build --full ns-control
dpkg-buildpackage: information: Quellpaket meta-mc
dpkg-buildpackage: information: Quellversion 1.0
dpkg-buildpackage: information: Quelldistribution unstable
dpkg-buildpackage: Information: Quelle geändert durch Frank Hofmann <frank.hofmann@efho.de>
dpkg-buildpackage: Information: Host-Architektur amd64
dpkg-source --before-build equivs.x2AVFx
fakeroot debian/rules clean
dh_testdir
dh_clean
```

Compatibility levels before 9 are deprecated (level 7 in use)
dpkg-source -b equivs.x2AVFx
dpkg-source: Information: Quellformat »1.0« wird verwendet
dpkg-source: Warnung: Quellverzeichnis »equivs.x2AVFx« lautet nicht <Quellpaket>-< ↳ Ursprungsversion> »meta-mc-1.0«
dpkg-source: Information: meta-mc wird in meta-mc_1.0.tar.gz gebaut
dpkg-source: Information: meta-mc wird in meta-mc_1.0.dsc gebaut
debian/rules build
make: Für das Ziel „build“ ist nichts zu tun.
fakeroot debian/rules binary
dh_testdir
dh_testroot
dh_prep
dh_testdir
dh_testroot
dh_install
dh_install: Compatibility levels before 9 are deprecated (level 7 in use)
dh_installdocs
dh_installdocs: Compatibility levels before 9 are deprecated (level 7 in use)
dh_installchangelogs
dh_compress
dh_fixperms
dh_installdeb
dh_installdeb: Compatibility levels before 9 are deprecated (level 7 in use)
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: Paket `meta-mc« wird in `../meta-mc_1.0_all.deb` gebaut.
dpkg-genbuildinfo
dpkg-genchanges `../meta-mc_1.0_amd64.changes`
...
$

Das Ergebnis umfaßt eine Reihe von Dateien und sieht dann wie folgt aus:

**Ergebnis nach dem Bauen des Pakets**

$ ls meta-mc*
meta-mc_1.0_all.deb  meta-mc_1.0_amd64.changes  meta-mc_1.0.tar.gz
meta-mc_1.0_amd64.buildinfo  meta-mc_1.0.dsc
$

**meta-mc_1.0_all.deb**
das erzeugte Metapaket

**meta-mc_1.0_amd64.buildinfo**
Dateien und Softwarepakete, die zum Zeitpunkt der Erstellung des Pakets installiert sind

$ cat meta-mc_1.0_amd64.buildinfo
Format: 1.0
Source: meta-mc
Binary: meta-mc
Architecture: all source
Version: 1.0
Checksums-Md5:
0e03c6890fc8a72762cf994a0cdcafb7 487  meta-mc_1.0.dsc
5050e909c14b6ccf703cf00ebf831594 2050  meta-mc_1.0_all.deb
Checksums-Sha1:
91ab2cf66937c606ff3abb88553f1555b78521a7d 487  meta-mc_1.0.dsc
c5640969e2e515a0fca2f49fc5835846e1d9a8fa 2050  meta-mc_1.0_all.deb
Checksums-Sha256:
ceed0cf69f3eeef5b4370b0c5654f0db3c5eb77809fc8057b79f91bd154d3c83 487  meta-mc_1.0.dsc
c2a1d26508cb413856c29afdf00e47b8850fa28a89c85510867b6a65644538 2050  meta-mc_1.0_all.deb
Build-Origin: Debian
Build-Architecture: amd64
Build-Date: Wed, 13 Nov 2019 14:55:05 +0100
Debian-Paketmanagement

Installed-Build-Depends:
  autoconf (= 2.69-10),
  automake (= 1:1.15-6),
  autopoint (= 0.19.8.1-2+deb9u1),
...
$

meta-mc_1.0_amd64.changes

die Änderungsichte zum erzeugten Paket. Diese Datei wird benötigt, um das erzeugte Paket später auf einem Paketmirror hochzuladen.

$ cat meta-mc_1.0_amd64.changes
Format: 1.8
Date: Wed, 13 Nov 2019 14:55:04 +0100
Source: meta-mc
Binary: meta-mc
Architecture: source all
Version: 1.0
Distribution: unstable
Urgency: low
Maintainer: Frank Hofmann <frank.hofmann@efho.de>
Changed-By: Frank Hofmann <frank.hofmann@efho.de>
Description:
  meta-mc - Installs the Midnight Commander
Changes:
  meta-mc (1.0) unstable; urgency-low

  * First version
Checksums-Sha1:
  91ab2cf66937c606ff3abb8853f1555b78521a7d 487 meta-mc_1.0.dsc
  68ae676fb11fcca32674557510327830499e17a2 1826 meta-mc_1.0.tar.gz
  c5640969e2e515afca2f49fc583586461d9a8fa 2050 meta-mc_1.0_all.deb
  42ece0cc919f89be786615c6fd5e0c992946455ca 5230 meta-mc_1.0_amd64.buildinfo
Checksums-Sha256:
  ced0cf69f3eeef5b437b0b5654f0db39eb77809fc8057b79f91db154d3c83 487 meta-mc_1.0.dsc
  27e3c42c64e1243371ff68bf62f255e863e54c8f5922326b5ddc494d8c3cb6cd 1826 meta-mc_1.0.tar.gz
  c2ca1265083bf413856c29afdfd00e47b8850fa28a89c85510867b6a65644538 2050 meta-mc_1.0_all.deb
  f94a055b0f2zaealf83e95debe3f842cf0a317f18cc8ff2a737025b33ad4e672 5230 meta-mc_1.0_amd64.buildinfo
Files:
  0e03c6890fc8a727626cf994a0cda3f7 487 misc optional meta-mc_1.0.dsc
  396f9392e9518bb302d88c4c038e095 1826 misc optional meta-mc_1.0.tar.gz
  5050e909c14b6cc7f703cf000eb831594 2050 misc optional meta-mc_1.0_all.deb
  261b4cb586d58d11a22badf6fb3914 5230 misc optional meta-mc_1.0_amd64.buildinfo
$

meta-mc_1.0.dsc

die Paketbeschreibung für die Paketverwaltung (Quellcode)

$ cat meta-mc_1.0.dsc
Format: 1.0
Source: meta-mc
Binary: meta-mc
Architecture: all
Version: 1.0
Maintainer: Frank Hofmann <frank.hofmann@efho.de>
Standards-Version: 3.9.2
Build-Depends: debhelper (>= 7)
Package-List:
Debian-Paketmanagement

meta-mc deb misc optional arch=all
Checksums-Sha1:
68ae676fb11fcca32674557510327830499e17a2 1826 meta-mc_1.0.tar.gz
Checksums-Sha256:
27e3c42c64e1243371ff68bf62f255e863e54c8f5922326b5ddc49d8c3cb6cd 1826 meta-mc_1.0.tar.gz
Files:
396fa6392e9518bb302d88c4c038e095 1826 meta-mc_1.0.tar.gz

Jetzt wurde das Paket erfolgreich gebaut — Gratulation!

22.3 Die Komponenten des Pakets kryptographisch signieren

Das zuvor gebaute Paket können Sie uneingeschränkt benutzen (und auch direkt bei Abschnitt 22.5 weiterlesen). In den nun folgenden Abschnitten gehen wir noch einen Schritt weiter in Richtung Sicherheit und versehen sowohl die einzelnen Komponenten des Pakets als auch das gesamte Paket selbst mit einer kryptographischen Signatur. Das garantiert jedem Nutzer die Echtheit des Pakets — sprich: der Inhalt und das Paket selbst stammt aus einer verlässlichen Quelle.


Signieren des Quellpakets

```bash
$ debsign -k D431AC07 meta-mc_1.0.dsc
signfile dsc meta-mc_1.0.dsc D431AC07
Successfully signed dsc file
$
```


Signieren der Änderungen

```bash
$ debsign -k D431AC07 meta-mc_1.0_amd64.changes
The .dsc file is already signed.
Would you like to use the current signature? [Yn]Y
Leaving current signature unchanged.
signfile buildinfo meta-mc_1.0_amd64.buildinfo
signfile buildinfo meta-mc_1.0_amd64.buildinfo D431AC07
fixup_changes dsc meta-mc_1.0.dsc meta-mc_1.0_amd64.changes
fixup_changes dsc meta-mc_1.0.dsc meta-mc_1.0_amd64.changes
signfile changes meta-mc_1.0_amd64.changes D431AC07
Successfully signed buildinfo, changes files
$
```


Prüfen der Signatur des Quellpakets

```bash
```
$ gpg --verify meta-mc_1.0.dsc

gpg: Signatur vom Do 14 Nov 2019 15:12:00 CET

gpg: mittels RSA-Schlüssel 35F8DF9C884E36AB974460AFCFA72978D431AC07

gpg: Korrekte Signatur von "Frank Hofmann (Hofmann EDV) <frank.hofmann@efho.de>" [unbekannt → ]

gpg: WARNUNG: Dieser Schlüssel trägt keine vertrauenswürdige Signatur!

gpg: Es gibt keinen Hinweis, daß die Signatur wirklich dem vorgeblichen Besitzer → gehört.

Haupt-Fingerabdruck = 35F8 DF9C 884E 36AB 9744 60AF CFA7 2978 D431 AC07
$

22.4 Das Debianpaket kryptographisch signieren

Für diesen abschließenden Schritt stehen zwei verschiedene Werkzeuge bereit—dpkg-sig (Debianpaket dpkg-sig [Debian-Paket-dpkg-sig]) und debsigs (Debianpaket debsigs [Debian-Paket-debsigs]). Leider sind deren Ergebnisse nicht zueinander kompatibel.

22.4.1 dpkg-sig verwenden

Rufen Sie dpkg-sig auf, erzeugt es eine Debian-control-Datei mit einer Reihe nützlicher Einträge:

- der Version von dpkg-sig, die die Signatur erzeugt hat
- die GnuPG-Information über denjenigen, in dessen Namen die Signierung vorgenommen wurde
- dessen Rolle
- einen Abschnitt mit Checksummen, Dateigrößen und Dateinamen der Binär-, Kontroll- und Datendateien im Debianpaket.

Der Name der erzeugten Datei hängt von der ausgewählten Rolle ab. Falls nicht weiter spezifiziert, ist das derjenige, der das Paket gebaut hat. Dann heißt die neue Datei _gpgbuilder. Die Datei wir dem Debianpaket hinzugefügt, das Paket wird mit GnuPG signiert und die Signatur als Klartext ergänzt.

Das Debianpaket signieren

$ dpkg-sig -k D431AC07 --sign builder meta-mc_1.0_all.deb
Processing meta-mc_1.0_all.deb...
gpg: "D431AC07" wird als voreingestellter geheimer Signaturschlüssel benutzt
Signed deb meta-mc_1.0_all.deb
$

Somit können Sie jetzt über mehrere Ebenen hinweg die Echtheit der Daten prüfen — entweder mittels gpg --verify, oder mittels dpkg-sig --verify.

Prüfen der Signatur mittels gpg

$ gpg --verify meta-mc_1.0_all.deb

gpg: Signatur vom Fr 15 Nov 2019 14:10:36 CET

gpg: mittels RSA-Schlüssel 35F8DF9C884E36AB974460AFCFA72978D431AC07

gpg: Korrekte Signatur von "Frank Hofmann (Hofmann EDV) <frank.hofmann@efho.de>" [unbekannt → ]

gpg: WARNUNG: Dieser Schlüssel trägt keine vertrauenswürdige Signatur!

gpg: Es gibt keinen Hinweis, daß die Signatur wirklich dem vorgeblichen Besitzer → gehört.

Haupt-Fingerabdruck = 35F8 DF9C 884E 36AB 9744 60AF CFA7 2978 D431 AC07
$

Prüfen der Signatur mittels dpkg-sig
$ dpkg-sig --verify meta-mc_1.0_all.deb
Processing meta-mc_1.0_all.deb...
GOODSIG _gpgbuilder 35F8DF9C884E36AB974460AFCFA72978D431AC07 1573823436
$

22.4.2 debsigs benutzen


Das Debiapaket signieren

$ debsigs --sign=origin -k D431AC07 meta-mc_1.0_all.deb

gpg: "D431AC07" wird als voreingestellter geheimer Signaturschlüssel benutzt
$

Nun ist auch der Sicherheitscheck über die Bühne und das Paket kann zum Einsatz kommen. Auf gehts!

22.5 Das neue Paket benutzen

Das fertige Paket steht nun zum Einsatz bereit. Entweder laden Sie dieses auf einen Paketmirror hoch (siehe Kapitel 30) oder Sie installieren das Paket gleich direkt auf Ihrem System. Für letzteres haben Sie mehrere Möglichkeiten.

22.5.1 Mittels dpkg und APT

Zuerst rufen Sie dpkg -i paket.deb auf, danach apt-get install -f. Ersteres veranlasst dpkg, ihr Paket zu installieren.

Installieren des Metapakets meta-mc mittels dpkg

# dpkg -i meta-mc_1.0_all.deb

Vormals nicht ausgewähltes Paket meta-mc wird gewählt.
(Lese Datenbank ... 239663 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von meta-mc_1.0_all.deb ... 
Entpacken von meta-mc (1.0) ...
dpkg: Abhängigkeitsprobleme verhindern Konfiguration von meta-mc:
  meta-mc hängt ab von mc; aber:
    Paket mc ist nicht installiert.
  meta-mc hängt ab von mc-data; aber:
    Paket mc-data ist nicht installiert.

dpkg: Fehler beim Bearbeiten des Paketes meta-mc (--install):
  Abhängigkeitsprobleme - verbleibt unkonfiguriert
Fehler traten auf beim Bearbeiten von:
  meta-mc
#

dpkg lässt das Paket unkonfiguriert, da es die im Paket benannten Paketabhängigkeiten nicht selbst auflösen kann. Sie erkennen den Status an der Buchstabenkombination iU für Paket installiert und nicht konfiguriert:

Installationsstatus von meta-mc

# dpkg -l | grep meta-mc
iU  meta-mc  1.0  all  ←
Installs the Midnight Commander
Nun rufen Sie `apt-get install -f` auf. APT behebt nun die fehlenden Abhängigkeiten und konfiguriert das Paket `meta-mc` wie folgt:

**Auflösen der Paketabhängigkeiten mit apt-get**

```bash
# apt-get install -f
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Abhängigkeiten werden korrigiert ... Fertig
The following additional packages will be installed:
 mc mc-data
Vorgeschlagene Pakete: 
 arj dbview djvulibre-bin gv libaspell-dev odt2txt python-boto python-tz
Die folgenden NEUEN Pakete werden installiert:
 mc mc-data
0 aktualisiert, 2 neu installiert, 0 zu entfernen und 2 nicht aktualisiert.
1 nicht vollständig installiert oder entfernt.
Es müssen noch 0 B von 1.780 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 7.175 kB Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/n] j
Vormals nicht ausgewähltes Paket mc-data wird gewählt.
(Lese Datenbank ... 239667 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von .../mc-data_3%3a4.8.18-1_all.deb ...
Entpacken von mc-data (3:4.8.18-1) ...
Vormals nicht ausgewähltes Paket mc wird gewählt.
Vorbereitung zum Entpacken von .../mc_3%3a4.8.18-1_amd64.deb ...
Entpacken von mc (3:4.8.18-1) ...
Trigger für mime-support (3.60) werden verarbeitet ...
Trigger für desktop-file-utils (0.23-1) werden verarbeitet ...
mc-data (3:4.8.18-1) wird eingerichtet ...
Trigger für man-db (2.7.6.1-2) werden verarbeitet ...
Trigger für hicolor-icon-theme (0.15-1) werden verarbeitet ...
mc (3:4.8.18-1) wird eingerichtet ...
meta-mc (1.0) wird eingerichtet ...
#
```

### 22.5.2 Mittels gdebi

Dem Werkzeug `gdebi` [Debian-Paket-gdebi] ist in Abschnitt 6.4.5 ein eigener Bereich gewidmet. Es untersucht das zuvor erzeugte Metapaket vor dessen Installation und löst die Paketabhängigkeiten sauber auf.

**Installation des Metapakets mittels gdebi**

```bash
# gdebi meta-mc_1.0_all.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading state information... Done
Erfordert die Installation folgender Pakete: mc mc-data

Installs the Midnight Commander
Installs the Midnight Commander
Wollen Sie das Software-Paket installieren? [j/N]:j
Fetched 0 B in 0s (0 B/s)
Vormals nicht ausgewähltes Paket mc-data wird gewählt.
(Lese Datenbank ... 240891 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von .../mc-data_3%3a4.8.18-1_all.deb ...
Entpacken von mc-data (3:4.8.18-1) ...
Vormals nicht ausgewähltes Paket mc wird gewählt.
Vorbereitung zum Entpacken von .../mc_3%3a4.8.18-1_amd64.deb ...
```
Entpacken von mc (3:4.8.18-1) ...
Trigger für mime-support (3.60) werden verarbeitet ...
Trigger für desktop-file-utils (0.23-1) werden verarbeitet ...
mc-data (3:4.8.18-1) wird eingerichtet ...
Trigger für man-db (2.7.6.1-2) werden verarbeitet ...
Trigger für hicolor-icon-theme (0.15-1) werden verarbeitet ...
mc (3:4.8.18-1) wird eingerichtet ...
Vormals nicht ausgewähltes Paket meta-mc wird gewählt.
(Lese Datenbank ... 241248 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von meta-mc_1.0_all.deb ...
Entpacken von meta-mc (1.0) ...
meta-mc (1.0) wird eingerichtet ...
#

22.5.3 Mittels apt

Das Werkzeug apt steht gdebi in nichts nach — es löst ebenfalls die Paketabhängigkeiten sauber auf. apt kann ebenfalls mit lokalen Paketen umgehen, sofern im Pfadnamen zum Paket ein Verzeichnistrenner enthalten ist. Liegt das Paket im lokalen Verzeichnis, fügen Sie vor dem Namen der Paketdatei die Zeichen / hinzu. Der Aufruf sieht dann wie nachfolgend gezeigt aus.

Abbildung 22.1: apt bei der Installation des Metapakets meta-mc
Kapitel 23

Paketformate mischen

23.1 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. tar.gz oder 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 daher „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 23.2).

23.2 Fremdformate mit alien hinzufügen

23.2.1 Einführung

Den Begriff alien übersetzen Sie am ehesten mit fremd, Fremdling oder Ausländer. alien heißt jedoch auch das gleichnamige Debianpaket [Debian-Paket-alien]. Es hilft Ihnen, eine Software, welche nicht als deb-Paket vorliegt, entsprechend umzuwandeln und für Ihre Debian-Installation vorzubereiten.


**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
die beschreiben, wie Sie das Paket wieder entfernen oder aktualisieren. Häufig befinden sich auch Hinweise dabei, die benennen, was dabei gegebenenfalls zu beachten ist.

Paketabhängigkeiten
die Informationen über Abhängigkeiten zu anderer Software.

Architektur oder Plattform
die Angabe der Prozessorarchitektur im Dateinamen, wie etwa amd64 bei deb-Paketen und x86_64 bei rpm-Archiven.

Diese Bestandteile benötigt die Paketverwaltung von Debian GNU/Linux, um die umgewandelten Softwarepakete korrekt in den Paketbestand einordnen zu können. Ohne die Informationen besteht bspw. keine Möglichkeit, die Abhängigkeiten zu anderen Softwarepaketen zu prüfen und sicherzustellen, dass alle benötigten Komponenten verfügbar sind und ggf. installiert werden können.

23.2.2 Pakete umwandeln

23.2.2.1 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 aus den Paketen [Debian-Paket-dpkg], [Debian-Paket-dpkg-dev] und [Debian-Paket-debhelper] zur Erzeugung von Debianpaketen
• den GNU-C-Compiler gcc und make ([Debian-Paket-gcc] und [Debian-Paket-make]), sofern Software im Quellcode vorliegt und aus diesem zu übersetzen ist


23.2.2.2 Durchführung

Die Umwandlung eines rpm-Pakets in ein deb-Paket erfolgt mit Hilfe des Werkzeugs alien, der Angabe des gewünschten Paketformats sowie über die originale Paketdatei. Das Paketformat geben Sie über den Schalter -d (Langform --to-deb) an. Nachfolgend demonstrieren wir Ihnen das anhand des Pakets mc für den Midnight Commander [mc]:

Umwandlung eines rpm-Pakets in ein deb-Paket

```
# alien -d mc-4.8.22-1.mga7.x86_64.rpm
mc-4.8.22-2_amd64.deb generated
```

Der Schalter -dist der von alien angenommene Standardfall und diesen können Sie somit im Aufruf weglassen. Darüberhinaus unterstützt alien noch die folgenden, weiteren Schalter zur Umwandlung zwischen anderen Formaten:

- \(r\) (Langform --to-rpm)
  Umwandlung in ein rpm-Paket

- \(t\) (Langform --to-tgz)
  Umwandlung in ein mit gzip komprimiertes tar-Archiv (Erweiterung .tar.gz), bspw. von der Linux-Distribution Slackware genutzt
--to-slp

-p (Langform --to-pkg)
Umwandlung zu (Open)Solaris-Paketen mit der Erweiterung .pkg (siehe [Solaris] und [OpenSolaris]).

Am Ende überprüfen Sie das Paket, um sicherzugehen, dass es auch tatsächlich zu ihrer bestehenden Installation passt. Dabei hilft Ihnen der Schalter -c (Langform --contents) von dpkg, um zu erkennen, an welche Stellen in ihrer Installation die einzelnen Komponenten des neuen Pakets installiert werden.

Wohin wird der Paketinhalt installiert

```
# dpkg --contents mc_4.8.22-2_amd64.deb
  drwxr-xr-x root/root 0 2019-11-16 12:11 ./
  drwxr-xr-x root/root 0 2019-11-16 12:11 ./etc/
  drwxr-xr-x root/root 0 2019-11-16 12:11 ./etc/profile.d/
   -rw-r--r-- root/root 153 2019-01-01 18:36 ./etc/profile.d/20mc.sh
   -rw-r--r-- root/root 49 2019-01-01 18:36 ./etc/profile.d/20mc.csh
  drwxr-xr-x root/root 0 2019-11-16 12:11 ./etc/mc/
   -rw-r--r-- root/root 21874 2019-01-01 18:35 ./etc/mc/mc.ext
   -rwxr-xr-x root/root 791 2018-12-28 20:35 ./etc/mc/edit.indent.rc
  ...
```

Mit einer Kombination aus den beiden Kommandos dpkg --info Paketname und grep Depends sehen Sie, welche Paketabhängigkeiten für das umgewandelte Paket bestehen und erfüllt sein müssen, damit dieses korrekt in ihr System eingespielt werden kann.

Abhängigkeiten von einem umgewandelten Paket

```
# dpkg --info mc_4.8.22-2_amd64.deb | grep Depends
Depends: libc6 (>= 2.15), libglib2.0-0 (>= 2.35.9), libgpm2 (>= 1.20.4), libslang2 (>= 2.2.4), libssh2-1 (>= 1.2.8)
```

Aus der obigen Ausgabe ersehen Sie, dass das Paket mc von den fünf Bibliotheken Libc, Glib2, Gpm2, Slang2 und Ssh2 abhängt.

### 23.2.2.3 Fallstricke und Besonderheiten bei der Umwandlung

Bei der Umwandlung des Paketformats bestehen mehrere Fallstricke. Diese führen regelmäßig dazu, dass die umgewandelte Software nicht wie erhofft funktioniert:

- die referenzierten Bibliotheken passen entweder gar nicht, nicht mehr (sind bspw. veraltet) oder sind nicht auffindbar. Bei letzterem sind diese bspw. nicht installiert oder haben schlicht und einfach einen anderen Paketnamen.
- die angegebenen Pfade im Originalpaket stimmen nicht mit Ihrer lokalen Verzeichnisstruktur oder der von Debian GNU/Linux genutzten Struktur überein. Hier helfen ggf. symbolische Links weiter, die dann auf das richtige Ziel verweisen.
- das Format der Konfiguration sorgt für Ärger, bspw. die verfügbaren Optionen und Schalter
- es bestehen Konflikte mit anderer, bereits installierter Software

Desweiteren haben deb-basierte Systeme zudem ihre Eigenheiten. Die folgenden Schalter von alien helfen Ihnen dabei, auch mit weiteren Sonderfällen bei der Umwandlung von Paketen klarzukommen:
--bump=Wert
ähnlich zu --version. Erhöhe die Versionsnummer des neuen Softwarepakets nicht um 1, sondern um den von ihnen im Aufruf angegebenen Wert.

--description=Beschreibung
Füge dem neuen Paket die genannte Beschreibung hinzu. Das ist insbesondere bei tar.gz-Dateien sinnvoll, da diese normalerweise noch keine Paketbeschreibung beinhalten.

--fixperms
bringe alle Angaben zu den Berechtigungen und den Eigentümern in Ordnung

--patch=Dateiname, --anypatch und --nopatch
automatisches Anpassen von Startup-Skripten und Pfaden gemäß dem File Hierarchy Standard (FHS)

--target=Architektur
setze die Plattform für das Paket auf den angegebenen Wert. Siehe Abschnitt 1.2 zu weiteren Angaben zur Plattform bzw. Architektur eines Paketes

--version=Versionsnummer
Füge dem neuen Paket die angegebene Versionsnummer hinzu. Das ist insbesondere bei tar.gz-Dateien sinnvoll, da diese normalerweise noch keine Versionsnummer beinhalten.

--veryverbose
noch ausführlicher als -v bzw. --verbose

-c (Langform --scripts)
erhalte die bestehenden Pre- und Post-Install- sowie Remove-Skripte eines Paketes

-g (Langform --generate) und --veryverbose
erweitern der Fehlersuche

-g (Langform --generate)
das Paket vor der Umwandlung noch bearbeiten. Der Schalter erzeugt ein Verzeichnis mit dem Paketinhalt und ermöglicht Ihnen damit die Ergänzung und Korrektur des Paketinhalts, bevor daraus ein neues Paket gebaut wird.

-k (Langform --keep-version)
die Versionsnummer des Paketes beibehalten. Normalerweise zählt alien diese bei der Umwandlung um eins hoch

-s (Langform --single)
wie -g, aber ohne das Verzeichnis packagename.orig zu erstellen. Der Schalter ist nützlich, wenn Sie ein Debianpaket erstellen möchten und zu wenig Speicherplatz zur Verfügung haben.

-T (Langform --test)
teste das erzeugte Debianpaket mit lintian [Debian-Paket-lintian] (siehe „Qualitätskontrolle“ in Kapitel 37).

-v (Langform --verbose)
aktiviere die ausführliche Ausgabe. alien gibt damit Informationen zu jedem einzelnen Schritt bei der Umwandlung eines Paketes an

-V (Langform --version)
Ausgabe der Version von alien

23.2.3 Umgewandelte Pakete einspielen

23.2.4 Pakete umwandeln und einspielen

In den obigen Schritten in Abschnitt 23.2.2 und Abschnitt 23.2.3 haben Sie zuerst ein Paket umgewandelt und danach installiert. alien kann jedoch auch beide Schritte in einem Rutsch durchführen — ein Paket von rpm nach deb umwandeln und danach gleich auf ihrem System einspielen. Dazu benutzen Sie den Schalter -i (Langform --install).

Das Paket paket.rpm mit alien umwandeln und einspielen

```
# alien -i paket.rpm
...
```

Anmerkung
Nach der Installation des Paketes löscht alien die lokal vorliegende Paketdatei.

23.2.5 Fazit


23.3 deb-Pakete in rpm-Strukturen

Auch für Linux-Distributionen, die auf dem rpm-Paketformat aufsetzen, können Sie deb-Pakete einspielen und benutzen. Es gelten dabei die gleichen Hinweise wie bereits unter "Fremdformate mit alien hinzufügen" Abschnitt 23.2 genannt — es ist nicht garantiert, dass das Paket in ihr System passt und erwartungsgemäß funktioniert.

Diese Möglichkeiten zur Umwandlung und Integration von deb-Paketen sind uns bekannt:

```
alien mit dem Schalter -r (Langform --to-rpm)
Umwandlung eines deb-Paketes in ein rpm-Paket. Haben Sie das Paket erfolgreich umgewandelt, installieren Sie es bspw. wie folgt mittels yum und dessen Schalter localinstall auf ihrem System:

Installation des umgewandelten Pakets mittels yum
```

```
# yum localinstall paket.rpm
...
```

Das Unterkommando localinstall veranlasst yum, nicht in den hinterlegten Paketquellen zu suchen, sondern das Paket aus dem lokalen Verzeichnis zu verwenden. Die Vorgehensweise ist vergleichbar mit dem Aufruf von dpkg -i.

Projekt apt4rpm [apt4rpm]
stammt im Original von Conectiva Linux [Conectiva]. Es erstellt aus einem APT-Repository ein lokales RPM-Repository, so dass die Pakete für yum verfügbar werden. Leider wird das Projekt seit 2012 nicht mehr weiterentwickelt.

Projekt apt-rpm [apt-rpm]

Projekt pacapt [Arch-Linux-pacapt]
Kapitel 24

Umgang mit LTS


Können Sie die bislang verwendete Veröffentlichung nicht wechseln, stellt sich die Frage, ob — und wenn ja wie — Sie die beste-hende Installation trotzdem noch weiter betreiben können. Variante 1 ist die Abschaltung der Gültigkeitsüberprüfung des Release Files. 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 (siehe Abschnitt 24.1 und Abschnitt 24.2). Variante 2 ist die Zuhilfenahme eines spezialisierten Dienstleisters, der sich um die Betreuung überfälliger LTS-Installationen im Rahmen des Projekts *Extended LTS* kümmert (siehe Abschnitt 24.3).

### 24.1 Kurzzeitiges Abschalten der Gültigkeitsüberprüfung des Release Files

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.

**Deaktivieren der Gültigkeitsüberprüfung des Release Files beim Aufruf von apt-get**

```
# apt-get -o Acquire::Check-Valid-Until=false update
```

Da Aptitude und *apt* den Schalter ebenfalls verstehen, können Sie den Aufruf somit auch mit diesen beiden Werkzeugen durchführen.

### 24.2 Dauerhaftes Abschalten der Gültigkeitsüberprüfung des Release Files

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]:

**Dauerhaftes Deaktivieren der Gültigkeitsüberprüfung des Release Files**

```
# echo "Acquire::Check-Valid-Until=\"False;\"" > /etc/apt/apt.conf.d/10no-check-valid-until
```
24.3 Dienstleister zur Pflege veralteter LTS-Installationen

Kapitel 25

Webbasierte Installation von Paketen mit 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.

25.1 Sinn und Zweck

Die Idee hinter apturl ist die Installation von Paketen aus den eingetragenen Repositories (bzw. Channels bei Ubuntu) über die Adresszeile Ihres Webrowsers. 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 Webbrowser und surfen die darüber genannte URL an:

```
apt://cowsay
```

apturl 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 lokal, eigenen Paketmirror (siehe Paketverwaltung beschleunigen in Kapitel 26).
25.2 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 8.32.1)

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.

25.3 apturl in der Praxis

- Todo:
  - Einsatzszenario
  - Empfehlungen zur Nutzung
Kapitel 26

Paketverwaltung beschleunigen

26.1 Hintergrund

• warum will man das:
  – Zeiteinsparung
  – 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 dauern auch länger (werden ausgehend, 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 36)

26.2 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 29
• debdelta, siehe Kapitel 42
• PDiffs (gibt’s nur bei Testing, Experimental und Sid), siehe auch Kapitel 42

• lokalen Zwischenspeicher für Pakete einrichten:
  – Proxy, der bereits herunterladene Pakete puffert (siehe Kapitel 28)
  – eigener Mirror (siehe Kapitel 31)

• weitere Softwarepakete zur Beschleunigung
  – *apt-fast* [apt-fast] (bisher existiert kein offizielles Debian-Paket, siehe auch [Debian-WNPP-RFP-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*

```sh
# apt-fast install cssed
...
```

### 26.3 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 8.43). 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 35).


• Räumen Sie auch den Paketcache auf. Das kann sowohl automatisiert, als auch manuell geschehen (siehe dazu Abschnitt 7.5).
Kapitel 27

Paketverwaltung hinter einem http-Proxy


27.1 Hintergrund

Es gibt unterschiedliche Gründe, warum Proxy Server überhaupt betrieben werden:


27.2 Varianten


Bezogen auf die Paketverwaltung ist der Spezialfall „Caching Proxy ausschließlich für Debianpakete“ interessant. Dieser behält die bereits angefragten Debianpakete in seinem Cache, um sie später schneller an weitere Clients im Netzwerk weitergeben zu können, sobald sie diese im Rahmen einer Softwareaktualisierung oder -installation anfragen. Damit beschleunigen Sie die

27.3 Einen Proxy konfigurieren


Im Debian-Ökosystem gibt es unterschiedliche Stellen, an denen Sie die Benutzung eines Proxy Servers hinterlegen. Dazu zählen zunächst die allgemeinen Umgebungsvariablen (siehe „Einen Proxy mittels Umgebungsvariablen konfigurieren“ in Abschnitt 27.4), danach folgen die Einstellungen des Desktop-Environments und der Anwendungen, wie bspw. im Webbrowser Firefox (siehe „Einen Proxy in Anwendungen konfigurieren“ in Abschnitt 27.5). Für die Paketverwaltung kommen die Einträge in der Datei /etc/apt/apt.conf bzw. der modularisierten Variante unter /etc/apt/apt.conf.d/70debconf ins Visier (siehe dazu „APT über HTTP-Proxy“ in Abschnitt 27.6).

27.4 Einen Proxy mittels Umgebungsvariablen konfigurieren

Viele Programme orientieren sich am Systemzustand und greifen dazu auf Umgebungsvariablen und lokale Konfigurationsdateien zurück. Zur Verwendung eines Proxy Servers sind diese Variablen üblich:

```
http_proxy
  HTTP Proxy Server

https_proxy
  HTTPS Proxy Server

ftp_proxy
  FTP Proxy Server

rsync_proxy
  Proxy Server für rsync [Debian-Paket-rsync]

no_proxy
  keinen Proxy Server verwenden
```

Nachfolgend sehen Sie beispielhaft, wie Sie auf der Kommandozeile eine solche Umgebungsvariable setzen und gleich systemweit zugänglich machen. Benutzt wird hier als Proxy Server die IP-Adresse 1.2.3.4 und als Port 8080.

Setzen einer Umgebungsvariable in der Shell

```
$ export http_proxy="http://1.2.3.4:8080/"
```

Ausführlichere Informationen zum Umgang mit Umgebungsvariablen finden Sie im Wiki von Arch Linux [proxyArch].

27.5 Einen Proxy in Anwendungen konfigurieren

Sie werden sich sicher fragen, was dieser Aspekt hier im Buch zu suchen hat. Wichtig wird das genau dann, wenn Sie hinter einem Proxy Server sitzen und dieser ihnen den direkten Weg zum Paketmirror oder auch zu dringend benötigten Informationsseiten versperrt. Dann wird die Recherche und auch der nachfolgende Bezug passender deb-Pakete nämlich etwas hakelig. Mehrfach waren wir in dieser Lage, was uns erstmal etwas ausgebremst hat. Wir fügen die Informationen dazu hier ein, damit Sie es leichter haben, aus einer solchen Situation herauszukommen, falls Sie unverhofft darin landen.
27.5.1 Mittels wget


Der nachfolgende Aufruf demonstriert, wie Sie das Paket `mc` [Debian-Paket-mc] via `wget` vom Paketmirror `ftp.de.debian.org` beziehen. Das heruntergeladene Paket finden Sie dann als Datei `mc_4.8.26-1.1_amd64.deb` in ihrem aktuellen Verzeichnis.

Bezug des Paketes `mc` vom Paketmirror `ftp.de.debian.org` via `wget`

```bash
$ wget http://ftp.de.debian.org/debian/pool/main/m/mc/mc_4.8.26-1.1_amd64.deb


Auflosen des Hostnamen »ftp.de.debian.org (ftp.de.debian.org)«... 141.76.2.4
Verbindungsauftbau zu ftp.de.debian.org (ftp.de.debian.org)|141.76.2.4|:80... verbunden.
HTTP-Anforderung gesendet, warte auf Antwort... 200 OK
Länge: 534020 (522K) [application/vnd.debian.binary-package]
In »mc_4.8.26-1.1_amd64.deb« speichern.
mc_4.8.26-1.1_amd64.deb 100%[==================================================================>] 521,50K 488 ←
KB/s in 1,1s
2023-06-12 22:07:04 (488 KB/s) - »mc_4.8.26-1.1_amd64.deb« gespeichert [534020/534020]
$```  

In Bezug auf Proxy Server verfügt `wget` über weitere Schalter, die sich im Alltag als recht nützlich erweisen:

```
--no-proxy-server

--proxy_user
   Authentifizierung beim Proxy Server mit dem angegebenen Benutzername.

--proxy_password
   Authentifizierung beim Proxy Server mit dem angegebenen Passwort.

Bitte beachten Sie bei der Verwendung der beiden Schalter `--proxy_user` und `--proxy_password` im direkten Aufruf über die Kommandozeile, dass die darüber angegebenen Authentifizierungsdaten in der Historie ihres Kommandointerpreters landen. Abhilfe schaffen Sie, indem Sie die Authentifizierungsdaten entweder in der lokalen Datei ~/.wgetrc oder systemweit in der Datei /etc/wgetrc hinterlegen.
```

27.5.2 Mittels curl


Bezug der Datei über einen Proxy mittels curl

```bash
curl --proxy "http://benutzer:passwort@127.0.0.1:1234" "http://webseite.org/datei"
curl --proxy "benutzer:passwort@127.0.0.1:1234" "http://webseite.org/datei"
```
Bitte beachten Sie beim direkten Aufruf über die Kommandozeile, dass die darüber angegebenen Authentifizierungsdaten in der Historie Ihres Kommandointerpreters landen. Abhilfe schaffen Sie auf drei möglichen Wegen. In **Variante 1** speichern Sie die benötigten Authentifizierungsdaten in den Umgebungsvariablen. curl wertet dazu `http_proxy` und `https_proxy` aus. Dann vereinfacht sich obiger Aufruf zu:

**Bezug der Datei über einen Proxy mittels curl (vereinfachter Aufruf)**

```bash
curl "http://webseite.org/datei"
```

In **Variante 2** speichern Sie die benötigten Authentifizierungsdaten in einer lokalen Datei und geben diese curl beim Aufruf als zusätzlichen Parameter mit. Das gelingt über den Schalter `-K` (Langform: `--config`), gefolgt vom Namen der Konfigurationsdatei. Der nachfolgende Aufruf benutzt die Datei `proxy-config` mit diesem Inhalt:

**Konfigurationsdatei für curl zur Verwendung eines Proxy Servers**

```bash
--proxy "http://benutzer:passwort@127.0.0.1:1234"
```

Der Aufruf von curl vereinfacht sich dann wie folgt:

**Bezug der Datei über einen Proxy Server mittels Konfigurationsdatei für curl**

```bash
$ curl -K proxy-config "http://webseite.org/datei"
```


Weitere Schalter zum Thema Proxy Server entnehmen Sie bitte der umfangreichen Manpage zu curl oder dem Curl Developer Guide [curl-developer](#).

### 27.5.3 Webbrowser Firefox

Firefox kann ebenfalls mit Proxy Servern umgehen. Eine Konfiguration dazu erlaubt Firefox jedoch nicht über Kommandozeilenschalter, sondern ausschließlich über den Menüpunkt Einstellungen → Allgemein → Netzwerkeinstellungen (siehe Abbildung 27.1 für das entsprechende Dialogfenster).

![Abbildung 27.1: Proxy-Einstellungen bei Firefox](#)

27.5.4 Webbrowser Chromium


Als Kommandozeilenparameter stehen Ihnen zur Verfügung:

```bash
--proxy-server
  den angegebenen Proxy Server benutzen. Für einen SOCKS Proxy v4 sieht der Aufruf wie folgt aus:
  $ chromium --proxy-server="socks4://webserver:1234"

--no-proxy-server
  keinen Proxy Server verwenden, auch wenn dieser systemweit konfiguriert ist

--proxy-auto-detect
  automatische Erkennung des Proxy Servers

--proxy-pac-url
  legt die URL der automatischen Konfiguration zur Erkennung des Proxy Servers fest
```

27.6 APT über HTTP-Proxy

27.6.1 Konfigurationsdateien und Einstellungen

Für `apt` tragen Sie den Proxy in der Datei `/etc/apt/apt.conf.d/70debconf` ein. Bei älteren Debian-Veröffentlichungen ist es die Datei `/etc/apt/apt.conf`. Nachfolgend sehen Sie den Eintrag für einen HTTP-Proxy namens `proxyserver`, der auf dem Port 8080 lauscht.

**Beispiel für einen Rechner proxyserver auf Port 8080**

```plaintext
Acquire::http::Proxy "http://proxyserver:8080";
```

Beispiel mit Zugangsdaten für Nutzer pakete

```
Acquire::http::Proxy "http://pakete:geheim@proxyserver:8080";
```


Abbildung 27.3: Proxy-Einstellungen bei Synaptic

### 27.6.2 Schalter zur Steuerung des Cache-Verhaltens

Wie APT mit den zwischengespeicherten Daten umgeht, legen Sie über mehrere Schalter fest.

**Cache-Start**

verfügbar ab Version 0.7.26. Legt die initiale Größe des Caches fest, die APT beim Start benutzt.

**Cache-Grow**

verfügbar ab Version 0.7.26. Legt die Schrittweite in Bytes fest, mit der der Cache vergrößert wird, sollte er nicht mehr ausreichend, 1048576 (ca. 1 MB). Die maximale Größe ist der Wert, der mittels **Cache-Limit** festgelegt. Falls als Wert 0 hinterlegt ist, ist die automatische Anpassung deaktiviert.

**Cache-Limit**

verfügbar ab Version 0.7.26. Legt die maximale Größe des Caches fest. Der Defaultwert ist 0 für keine Größenbeschränkung.

**Max-Age**

Alter der Indexdatei in Sekunden
No-Cache
unter keinen Umständen die zwischengespeicherten Inhalte verwenden

No-Store
angefragte Daten nicht im Cache zwischenspeichern

27.6.3 Umgebungsvariablen

APT greift auf Umgebungsvariablen zurück, sofern Sie diese auf Ihrem System definiert haben. Es benutzt ftp_proxy, http_proxy und https_proxy. Wie Sie diese festlegen, lesen Sie in Abschnitt 27.4 nach.

27.6.4 Schalter für apt-get


Beispielaufruf zur Installation von mc via Proxy

```
# apt-get -o http::Proxy="http://proxyserver:8080" install mc
```

Die vollständige Angabe für Proxy Server folgt dieser Form:

http://[[Benutzername][:Passwort]@]Rechner[:Port]/

Bitte beachten Sie, dass bei obigem Aufruf die verwendeten Zugangsdaten aus Benutzername und Passwort im Klartext in der Historie ihres Kommandozeileninterpreters abgelegt werden.

27.7 Aptitude über HTTP-Proxy

Für Aptitude gilt das Gleiche wie bei APT. Hintergrund ist, dass sowohl APT, als auch Aptitude die gleiche Bibliothek namens libapt benutzen. Somit sind das Featureset und die Konfiguration beider Werkzeuge an dieser Stelle identisch (siehe Abschnitt 27.6).
Kapitel 28

Einen APT-Paket-Cache einrichten

28.1 Begriff

Im Allgemeinen bezeichnet ein Cache einen Datenpuffer oder Zwischenspeicher, der Daten weiter vorhält und auf den im Bedarfsfall zurückgegriffen wird, wenn die gleichen Daten erneut angefordert oder ausgetauscht werden. In unserem speziellen Fall beinhaltet der Cache keine beliebigen Daten, sondern Softwarepakete. Diese werden darin von der Paketverwaltung gespeichert, sobald sie das erste Mal vom Paketmirror bezogen wurden. Das passiert beispielsweise bei der Installation oder Aktualisierung eines Softwarepakets.


28.2 Besonderheiten des APT-Cache

28.2.1 Funktionsweise

28.2.2 Vor- und Nachteile

Als vorteilhaft sehen wir zunächst den geringeren Bedarf an Speicherplatz an — die Softwarepakete werden nur ein einziges Mal im gesamten Rechnerverbund vorgehalten und nicht mehr auf jedem einzelnen Knoten im Netzwerk. In Folge verringert sich die Anzahl der lokalen Lese- und Schreibzugriffe auf den Speichermedien der einzelnen Knoten und trägt somit dazu bei, deren Lebensdauer zu verlängern.


Der APT-Cache läßt sich auch als Element zur Erhöhung der Ausfallsicherheit des Netzwerks ansehen: fällt der externe Paketmirror oder gar der Internetzugang aus, stehen ihnen immer noch die bislang bezogenen Softwarepakete zur Verfügung, die sich bereits im APT-Cache befinden. Bereits begonnene Aktualisierungen des Paketbestands auf anderen Knoten im Netzwerk lassen sich fortsetzen und auch vollenden, sofern sich alle dazu benötigten Pakete bereits im APT-Cache befinden.


28.2.3 Abgrenzung zum Betreiben eines eigenen Paketmirrors

Vor der Einrichtung wägen Sie bitte zwischen einem APT-Cache und einem eigenen Paketmirror ab (siehe Kapitel 31). Während letzterer stets alle Softwarepakete der Veröffentlichung vorhält — also auch die, die Sie nicht benötigen — landen im APT-Cache hingegen nur die Pakete, die bisher tatsächlich von mindestens einem der Knoten im Netzwerk angefragt wurden. Ein APT-Cache ist daher auch schlanker als ein eigener Paketmirror.
Zusätzliche Zeit ist hingegen einmalig einzuplanen, wenn Softwarepakete von einem externen Paketmirror bezogen werden, weil diese sich dann ja noch nicht im APT-Cache befinden. Das trifft insbesondere dann zu, wenn neue Paketversionen auf dem Paketmirror bereitstehen und ein Knoten im Netzwerk diese neue Version anfordert.

Vom Betreiben eines eigenen Paketmirrors mit vorgeschaltetem APT-Cache im gleichen Netzwerk raten wir Ihnen ab. Wir sehen in dieser Kombination von Diensten keinen wesentlichen Leistungsgewinn.

28.2.4 Softwareauswahl für einen APT-Cache

Nachfolgend stellen wir Ihnen unterschiedliche Lösungen vor, mit denen Sie einen APT-Cache einrichten und betreiben können. Welche Lösung für Sie am besten passt, hängt von ihrer konkreten Situation ab.

28.3 Approx

28.3.1 Überblick

Approx [Debian-Paket-approx] ist ein Proxyserver für Debianpakete und dient als Paketcache zwischen einem Paketmirror und Debian-basierten Clients. Mit Approx halten Sie Debianpakete vor, die von den Clients zur lokalen Installation oder Aktualisierung angefragt werden und liefern diese aus dem Paketcache an die Clients aus.

Gemäß der Selbstbeschreibung von Approx sind lediglich die Repositorys in den Konfigurationsdateien von Approx anzupassen, jedoch nicht die Konfigurationsdateien jedes Clients selbst (bspw. /etc/apt/sources.list). In der Realität sieht es jedoch etwas anders aus und es sind ebenfalls Änderungen in den Clients notwendig, damit diese mit dem Paketproxy kommunizieren (siehe Abschnitt 28.3.3.2).

Approx präsentiert sich als Alternative zu APT Cacher (siehe Abschnitt 28.4) und APT Cacher-NG (siehe Abschnitt 28.5), dem Nachfolger von APT Cacher.

28.3.2 Setup und Installation

Als ersten Schritt laden Sie das Paket approx [Debian-Paket-approx] herunter und installieren es, bspw. mittels apt-get wie folgt:

Installation von Approx mit apt-get

```
apt-get install approx
```

Das Paket approx ist rund 1.5 MB groß und belegt nach dem Entpacken des Softwarearchivs etwa 6.6 MB Platz in ihrem System. Weitere Pakete sind zur Installation nicht erforderlich. Die Schritte 2 und 3 beinhalten nun die Konfiguration des Approx-Serverdienstes (siehe Abschnitt 28.3.3.1) und der Clients (siehe Abschnitt 28.3.3.2).

28.3.3 Konfiguration

28.3.3.1 Server


$cache
legt den Pfad fest, den Approx für seinen eigenen Paketcache verwendet. Der Defaultwert ist /var/cache/approx.

$curl_path
legt den Pfad zum Programm curl fest. Der Defaultwert ist /usr/bin/curl.
$debug
legt fest, ob Logeinträge zur erweiterten Fehlersuche erstellt werden sollen. Der Defaultwert ist false und somit ausgeschaltet.

$group
legt den Namen der Gruppe fest, dem die Dateien im Cacheverzeichnis von Approx gehören. Der Defaultwert ist approx.

$interval
gibt die Zeitdauer in Minuten an, nach der eine Datei im Cache als zu alt angesehen wird, um noch sofort von Approx ausgeliefert zu werden. Ist das Intervall überschritten, schaut Approx zuerst auf dem Paketmirror nach, ob es zwischenzeitlich eine neuere Version gibt. Der Defaultwert ist 60 für 60 Minuten.

$max_rate
legt die Obergrenze für die Breite des Datenstroms in Bytes pro Sekunde fest, mit der Daten vom Paketmirror bezogen werden. Der Defaultwert ist unlimited für unbegrenzte Bandbreite. Geben Sie stattdessen einen Zahlenwert an und ergänzen ihn mit K, M oder G, begrenzen Sie die Bandbreite auf den entsprechenden Wert in Kilobyte, Megabyte oder Gigabyte pro Sekunde.

$max_redirects
legt die maximale Anzahl von HTTP-Weiterleitungen (HTTP redirects) fest, denen gefolgt wird, wenn ein Paket heruntergeladen wird. Der Defaultwert ist 5.

$max_wait
gibt an, wie viele Sekunden der Prozess von Approx im Falle eines konkurrierenden Downloads wartet, bevor es die Datei selbst bezieht. Der Defaultwert ist 10 für 10 Sekunden.

$offline
gibt an, ob Approx auch möglicherweise veraltete Dateien ausliefern soll, wenn keine Verbindung zum Paketmirror zustande kommt. Der Defaultwert ist false und diese Funktionalität ist somit ausgeschaltet.

$pdiffs
legt fest, ob IndexFile Diffs genutzt werden sollen. Der Defaultwert steht auf true für eingeschaltet.

$syslog
legt das Verhalten von syslog fest, wenn mit Logdateien umgegangen wird. Der Defaultwert ist daemon.

$user
legt den Namen des Benutzers fest, dem die Dateien im Cacheverzeichnis von Approx gehören. Der Defaultwert ist approx.

$verbose
legt fest, ob ausführlichere Logeinträge erstellt werden sollen. Der Defaultwert ist false und somit ausgeschaltet.


Nachfolgend sehen Sie in beispielhaft die Einstellungen für einen Paketmirror, welcher die Softwarepakete unter /var/cache/paket zwischenpuffert.

**Beispielkonfiguration für Approx**

```
debian http://deb/debian.org/debian
security http://security.debian.org/debian-security

$interval 60
$max_wait 10
$cache /var/cache/paketcache
```

Ohne weitere Änderungen lauscht Approx auf dem Port 9999 für Anfragen von Clients. Um das zu ändern und Approx bspw. auf Port 8000 lauschen zu lassen, passen Sie die Servicebeschreibung von Approx an. Verwenden Sie Systemd als Prozessmanager, hilft ihnen dabei das Kommando systemctl weiter.

**Anpassen der Servicebeschreibung von Approx bei Systemd**
systemctl edit approx.socket

Die Servicebeschreibung beinhaltet die drei Abschnitte Unit, Socket und Install. Ändern Sie im Abschnitt Socket die Einstellung ListenStream=9999 auf ListenStream=8000.

**Veränderte Servicebeschreibung von Approx bei Systemd**

```
[Unit]
Description=caching proxy server for Debian archive files
Documentation=man:approx(8)

[Socket]
ListenStream=8000
Accept=yes

[Install]
WantedBy=sockets.target
```

Verwenden Sie hingegen **Init.d** als Prozessmanager, ändern Sie zunächst in der Konfigurationsdatei `/etc/inetd.conf` diese Zeile von

<table>
<thead>
<tr>
<th>9999</th>
<th>stream</th>
<th>tcp</th>
<th>nowait</th>
<th>approx /usr/sbin/approx /usr/sbin/approx</th>
</tr>
</thead>
</table>

in

<table>
<thead>
<tr>
<th>8000</th>
<th>stream</th>
<th>tcp</th>
<th>nowait</th>
<th>approx /usr/sbin/approx /usr/sbin/approx</th>
</tr>
</thead>
</table>

und speichern die Konfigurationsdatei. Danach starten Sie Approx wie folgt neu:

`/etc/init.d/approx restart`

Danach passen Sie die Konfiguration des Clients an—dieser soll ja jetzt auch mit dem APT-Cache kommunizieren und nur darüber seine Softwarepakete beziehen.

### 28.3.3.2 Client

Die Konfiguration des Clients passen Sie in drei Schritten an. **Schritt 1** ist die Ergänzung der Datei `/etc/hosts` um den APT-Cache mit der entsprechenden IP-Adresse und dem Hostname. Heißt ihr APT-Cache *skye* und hat die IP-Adresse 192.168.10.250, fügen Sie diesen Eintrag zur Datei hinzu:

```
192.168.10.250 skye
```

In **Schritt 2** ergänzen Sie in der Datei `/etc/apt/sources.list` die Einträge der Paketmirror um den Verweis auf den zuvor angelegten APT-Cache.

```
deb http://skye:8000/debian stable main contrib non-free
deb-src http://skye:8000/debian stable main contrib non-free
deb http://skye:8000/security stable/updates main contrib non-free
```


### 28.3.4 Beobachtungen aus dem Alltag

- Verzögerungen einplanen, bis Pakete im Approx-Cache gelandet sind
28.4 apt-cacher

28.4.1 Werkzeugbeschreibung


Netztechnisch klemmt sich die Software zwischen dem (öffentlichen) Paketmirror und den anfragenden Clients. Es ist interessant für kleine und auch größere Netzwerke, die einen weitestgehend identischen Softwarebestand besitzen. apt-cacher ermöglicht einen Paketcache, d.h. es merkt sich, welche Pakete bereits angefragt wurden und liefert diese an den Anfragenden aus dem Paketcache, sofern sich das Paket bereits darin befindet, ansonsten holt es das Paket zuvor vom Paketmirror.

28.4.2 Einrichtung und Benutzung

Die Einrichtung unterteilt sich in zwei Schritte — die Konfiguration des Servers und der Clients. Der Server ist hierbei der Rechner, der den Paketcache später bereitstellt.

28.4.2.1 Serverkonfiguration

Zunächst installieren Sie das Paket apt-cacher auf dem Server, idealerweise in Kombination mit einem APT-Proxy wie apt-proxy [apt-proxy] oder approx (siehe Abschnitt 28.3). Danach aktivieren Sie den Dienst zum automatischen Start von apt-cacher, indem Sie noch eine kleine Änderung in der Konfigurationssatei /etc/default/apt-cacher vornnehmen. Ändern Sie darin diese Zeile wie folgt von

```
`AUTOSTART`: `AUTOSTART=0`
```

in

```
`AUTOSTART`: `AUTOSTART=1`
```

Nachdem Sie die Konfigurationsdatei gespeichert haben, starten Sie apt-cacher neu, bspw. mittels /etc/init.d/apt-cacher restart.

28.4.2.2 Client anpassen

Auf einem Clients ändern Sie nun die Bezugsadresse für Softwarepakete. Dazu legen Sie einen neuen Service an — die Datei namens /etc/apt/apt.conf.d/90-apt-proxy.conf. Diese Datei ergänzen Sie um den folgenden Eintrag, indem Sie damit auf ihren APT-Cache verweisen:

```
Acquire::http::Proxy "http://repository-cache:3142";
```

Hier bezeichnet repository-cache den von uns beispielhaft ausgewählten Rechnernamen, der den Paketcache bereitstellt. Die Angabe 3142 bezeichnet den Port, auf dem der Paketcache auf die Anfragen des Clients lauscht.

28.5 apt-cacher-ng

28.5.1 Werkzeugbeschreibung


apt-cacher-ng sieht sich selbst als Proxyserver zum Zwischenspeichern von Softwaredepots und lädt Pakete von Softwaredepots im Debian-Stil (oder möglicherweise von anderen Typen) herunter. Es wurde von Grund auf neu als Ersatz für apt-cacher entwickelt (siehe Abschnitt 28.4) und soll gleichzeitig apt-proxy [apt-proxy] und approx (siehe Abschnitt 28.3) ersetzen, ohne die Datei /etc/apt/sources.list der Clients ändern zu müssen.

Im laufenden Betrieb speichert apt-cacher-ng eine Kopie aller Nutzdaten, die über ihn laufen.
28.5.2 Einrichtung und Betrieb

• ToDo
Kapitel 29

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. 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.

29.1 Paketarchiv als tmpfs-Partition

In Schritt eins überdenken Sie dazu die Größe der Partition. Wägen Sie dabei zwischen dem potentiellen 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?


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/
```

`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)

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.


### 29.2 Paketcache als separate Partition einrichten


### 29.3 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
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

```bash
$ 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";
```


### Symlink zum neuen Paketcache anlegen

```bash
# 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.

---

²Im Normalfall sollte sogar der Symlink alleine auch ausreichen, damit APT einen alternativen Cache findet. APT folgt dann einfach auch dem Symlink.
Kapitel 30
Eigenes APT-Repository anlegen

Wie Sie in den vorhergehenden Abschnitten gesehen haben, hängt es von verschiedenen Faktoren ab, ob ein Paket Bestandteil der offiziellen Veröffentlichung von Debian wird. Benötigen Sie eine bestimmte Software häufiger, die noch nicht oder auch nicht mehr in einem offiziellen Repository enthalten ist, ist das Anlegen eines eigenen APT-Repositorys durchaus sinnvoll. Damit verringern Sie die Anzahl der Arbeitsschritte zur Pflege ihres Softwarebestands erheblich. Als Motivation für eigene APT-Repositories sehen wir:

- das Bereitstellen einer lokalen Paketauswahl, sprich: einfach nur Pakete mit einem bestimmten Zweck oder Funktionsbereich vorhalten, bspw. für eine eigene Distribution (quasi ein unvollständiger Paketmirror)
- das Erstellen eigener, bislang nicht paketierter Software
- das Neubauen bestehender Software als Paket und dessen lokale Bereitstellung, bspw. mit Optimierung für die tatsächlich genutzte Hardware bzw. Architektur

Die nachfolgend vorgestellten Lösungsvarianten sind recht unterschiedlich und reichen von einem schlichten Verzeichnis bis hin zu einem eigenen Paketmirror, den Sie auch später offiziell referenzieren können.

30.1 Verzeichnis mit Paketen

Diese Variante ist aus technischer Sicht sehr simpel. Im 1. Schritt legen Sie ein Verzeichnis fest, welches Sie als Ablage für die *.deb-Pakete benutzen möchten. Sofern noch nicht vorhanden, legen Sie dieses an, bspw. wie folgt:

**Schritt 1: Verzeichnis für das Paketarchiv anlegen**

```
# mkdir -p /opt/paketarchiv
```

Der Schalter `-p` sorgt dafür, dass der vollständige Pfad `/opt/paketarchiv` angelegt wird. Damit umgehen Sie die Situation, dass `mkdir` seine Ausführung abbricht, falls eine von Ihnen im Pfad angegebene Verzeichnisebene noch nicht existiert.

Im 2. Schritt kopieren Sie alle gewünschten *.deb-Pakete in dieses Verzeichnis, bspw. mit Hilfe von diesem Kommando:

**Schritt 2: Daten hineinkopieren**

```
# cp -v *.deb /opt/paketarchiv/.
```


Benötigen Sie nun bspw. das Paket *software-0.2.3* aus Ihrem Archiv, installieren Sie dieses nun mit Hilfe von `dpkg` und mit dem Verweis auf ihr Paketarchiv wie folgt:

**Paket mittels dpkg aus dem Paketarchiv installieren**

```
# dpkg -i /opt/paketarchiv/software-0.2.3.deb
```
Um dieses Paketarchiv zu pflegen, genügt es vollkommen, nicht mehr benötigte Pakete aus dem Verzeichnis zu löschen und neue oder aktualisierte Pakete in dieses Verzeichnis zu kopieren. Mehr ist dafür nicht zu tun.

### Anmerkung

### 30.2 dpkg-scanpackages


#### Schritt 1: Verzeichnis für das Paketarchiv anlegen

```bash
# mkdir -p /opt/paketarchiv/local
```

Im 2. Schritt kopieren Sie alle gewünschten .deb-Pakete in dieses Verzeichnis:

#### Schritt 2: Daten hineinkopieren

```bash
# cp -v *.deb /opt/paketarchiv/local
```


Nun erstellen Sie die benötigten Metadaten mittels dpkg-scanpackages. Als minimale Angabe erwartet dpkg-scanpackages den Pfad zum Paketarchiv und schreibt dann seine Ausgabe auf stdout. Besser ist es, die Ausgabe mittels Umleitung operator > gleich in eine passende Datei namens Packages umzuleiten (Schreibweise des Dateinamens mit großem P). Der nachfolgende Aufruf geht noch einen Schritt weiter und komprimiert die Ausgabe zusätzlich mittels gzip, so dass als Ergebnis die Datei Packages.gz entsteht:

#### Schritt 3: Metadaten zum Paketarchiv erzeugen (Packages.gz)

```bash
# cd /opt/paketarchiv
# dpkg-scanpackages local/ | gzip -c > local/Packages.gz
```

Liegen bspw. 6 Debian-Pakete in Form von .deb-Dateien im Paketarchiv, liest dpkg-scanpackages deren Metadaten, gibt eine Meldung wie oben aus und leitet die Informationen an gzip weiter. gzip komprimiert die empfangenen Daten zunächst und speichert diese dann als Datei Packages.gz im Verzeichnis des Paketarchivs. Die erzeugte Datei Packages.gz beinhaltet Informationen pro Paket in der folgenden Form:

#### Paketinformationen zum beispielhaft verwendeten Paket meta-mc

```bash
# cat local/Packages
Package: meta-mc
Version: 1.0
Architecture: all
Maintainer: Frank Hofmann <frank.hofmann@efho.de>
Installed-Size: 9
Depends: mc, mc-data
```

Anmerkung
Bitte beachten Sie, dass Sie diesen Schritt mittels `dpkg-scanpackages` jedes Mal wiederholen müssen, wenn Sie Pakete im Paketarchiv hinzufügen oder Pakete daraus entfernen. Mit dem folgenden Shellskript lässt sich der Schritt automatisieren:

**Automatisierung mit Hilfe eines Shellskriptes**

```bash
#!/bin/bash
dpkg-scanpackages /opt/paketarchiv/local | gzip -c > /opt/paketarchiv/local/Packages.gz
```

Idealerweise legen Sie das Skript unter einen passenden Pfad wie bspw. `/usr/bin` ab. Damit vereinfacht sich ihr Aufruf.

Nun folgt der 4. Schritt — das Hinzufügen ihres soeben erzeugten, lokalen Paketarchivs zur Liste der von Ihnen genutzten Repositorys (siehe Abschnitt 3.3). Entweder ergänzen Sie die Datei `/etc/apt/sources.list` um eine Zeile mit Ihrem Paketarchiv oder Sie erstellen eine separate Datei im Verzeichnis `/etc/apt/sources.list.d`. Für obiges Beispiel sieht die Zeile wie folgt aus, um Ihr Paketarchiv zunächst lokal einzubinden:

**Schritt 4: Zusätzlicher Eintrag in der Datei `/etc/apt/sources.list`**

```bash
deb [trusted=yes] file:///opt/paketarchiv local/
```


Um das Paketarchiv nun auch für andere Benutzer via HTTP oder HTTPS zugänglich zu machen, stellen Sie das Verzeichnis bspw. mit Hilfe eines lokalen Webservers bereit. Ein gültiger Eintrag wäre bspw. der folgende:

**Zusätzlicher Eintrag in der Datei `/etc/apt/sources.list` (Webserver)**

```bash
deb [trusted=yes] https://www.webserver.com/paketarchiv local/
```

Anmerkung


Anmerkung
Bitte beachten Sie, dass es bei dieser Lösung keine Authentifizierung und Integritätsprüfung der Pakete im Archiv gibt. Die Verantwortung über die bereitgestellten Softwarepakete liegt vollständig bei Ihnen.
30.3 reprepro

reprepro [Debian-Paket-reprepro] ist ein Programm, das APT-Repositories aus unterschiedlichen Quellen bauen kann, u.a. als Mirror oder aus lokalen Binär- oder Quellpaketen.

TODO:
• Querverweise zum Aufsetzen eines eigenen Mirrors
• Die unterschiedlichen Unterkommandos.
• Wie werden verschiedene Distributions-Releases verwaltet
• Handhabung von reinen Quell- oder Binär-Paketen als auch .changes-Dateien.
• gpg-Schlüssel zum Signieren der bereitgestellten Pakete

30.4 mini-dinstall


30.4.1 Voraussetzungen


30.4.2 Einrichtung

30.4.2.1 Verzeichnisstruktur des APT-Repositorys


Verzeichnis für das APT-Repository anlegen

```bash
# mkdir -vp /var/www/debian
mkdir: Verzeichnis '/var/www/debian' angelegt
#
```

Danach legen Sie ein Unterverzeichnis an, indem die neuen, von Ihnen später hochgeladenen Pakete zunächst landen. In unserem Fall heißt dieses lediglich neupakete:

Verzeichnis für neue Pakete anlegen

```bash
# mkdir -vp /var/www/debian/neupakete
mkdir: Verzeichnis '/var/www/debian/neupakete' angelegt
#
```
30.4.2.2 Konfiguration für mini-dinstall

Als nächstes legen Sie eine Konfigurationsdatei für mini-dinstall an. Der übliche Name dafür ist .mini-dinstall.conf in dem Home-Verzeichnis des Benutzers, der die Pakete später in das APT-Repository hochlädt. Eine einfache Konfigurationsdatei .mini-dinstall.conf sieht wie folgt aus:

**Basiseinstellungen zu .mini-dinstall.conf**

```
[DEFAULT]
archivedir = /var/www/debian
archive_style = flat
architectures = all, amd64
mail_on_success = false
```

Damit liegt das APT-Repository unter /var/www/debian mit einer flachen Struktur (alle Binärpakete landen im gleichen Unterverzeichnis) und Unterverzeichnisse werden nur für die beiden Architekturen amd64 und all angelegt. Es erfolgt keine Benachrichtigung bei einem erfolgreichen Upload. Die Einstellungen gelten für alle Archive (Abschnitt [DEFAULT]), spezifische Einstellungen pro Veröffentlichungsstatus bestehen nicht.

Mit obigen Angaben ist ihr APT-Repository bereits benutzbar. APT und aptitude erwarten jedoch noch zusätzliche Informationen, um die Pakete in ihrem eigenen Repository auch als vertrauenswürdig anzuerkennen. **Schritt 1 ist eine Release-Datei.** Dafür ergänzen Sie die folgende Zeile am Ende der obigen Datei .mini-dinstall.conf:

**Ergänzungen zu .mini-dinstall.conf**

```
generate_release = 1
```

Laden Sie später ein Paket hoch oder rufen Sie mini-dinstall direkt auf, erzeugt es eine solche Release-Datei mit den Hashwerten der Pakete in ihrem Repository. Diese hat dann bspw. den folgenden Inhalt:

**Inhalt der Release-Datei**

```
# cat /var/www/debian/unstable/Release
Origin: root
Label: root
Suite: unstable
Codename: unstable
Date: Wed, 13 Nov 2019 17:02:23 UTC
Architectures: all amd64
MD5Sum:
d41d8cd98f00b204e9800998ecf8427e 0 Packages
f88f7e4006f620fe9eb58d8db16b863c3 33 Packages.gz
4059d198786f98dc9372dc1c54bc3c3 14 Packages.bz2
d41d8cd98f00b204e9800998ecf8427e 0 Sources
e86c520b4bec9c320a71529f9f2275f 32 Sources.gz
4059d198786f98dc9372dc1c54bc3c3 14 Sources.bz2
SHA1:
da39a3ee5e6eb4b0d3255bfe9f5601890af08190 0 Packages
91199c612d2cf161208659f6218378c86498df7e 33 Packages.gz
64a543afbbf54bf72863bdcbebe7a2ed0804adcd2 14 Packages.bz2
da39a3ee5e6eb4b0d3255bfe9f5601890af08190 0 Sources
17beadd3c12d3ce8079e27fc4a7f31f203c5705 32 Sources.gz
64a543afbbf54bf72863bdcbebe7a2ed0804adcd2 14 Sources.bz2
SHA256:
e3b0c42298f7c1c149afbf4c8996fb2d27ae41e4649b934ca495991b7852b855 0 Packages
adfd9c843936b6d4263f0ae73f49ca71b98765143f3d76d62182adaf7f25dc 33 Packages.gz
9d3dab4e03b9738d1118e2b2e7e8e24610690049c30ae07819ad60815134a8058 14 Packages.gz
2340be86a71b7f6c0ed90857b3828d19334bf03693cb846114f8277b09facaf 32 Sources.gz
3d3da84eb03b9738d1118e2b2e7e8e24610690049c30ae07819ad60815134a8058 14 Sources.gz
```

SHA256:
**Anmerkung**

### 30.4.2.3 Konfiguration für dput


Ein üblicher Name für die Konfigurationsdatei ist .dput.cf im Home-Verzeichnis des Benutzers, der die Pakete später in das APT-Repository hochlädt. Darin stellen sie dput passend zum weiter oben genannten APT-Repository ein, bspw. wie folgt:

**.dput.cf-Beispiel:**
```
[local]
fqdn = localhost
method = local
incoming = /var/www/debian/neuepakete
run_dinstall = 0
post_upload_command = mini-dinstall -b
```


### 30.4.2.4 Paket in das Repository hochladen


Der Schalter -u im Aufruf kürzt --unchecked ab und überspringt den Test der kryptographischen Signaturen zum Paket. Das ist zum Testen ganz praktisch.

**Hochladen eines Paketes mit dput**
```
# dput -u local /home/frank/projekte/metapackage/meta-mc_1.0_amd64.changes
Uploading to local (via local to localhost):
Successfully uploaded packages.
```

Benutzen Sie kein dput oder dput-ng, laden Sie alle Dateien, die in der .changes-Datei zum Paket benannt sind, in das Unterverzeichnis, indem neue Pakete landen (dput macht nichts anderes).

- Einrichtung (was noch schön wäre)
  - gpg-Schlüssel zum Signieren der bereitgestellten Pakete

### 30.4.3 Laufender Betrieb

- APT-Repository zu Ihren Paketquellen hinzufügen
  - neuer Eintrag in der /etc/apt/sources.list
Debian-Paketmanagement

# local repo
deb file:/var/www/debian/ unstable/
deb-src file:/var/www/debian/ unstable/

- apt-get update
- Ergänzungen
  - cron oder daemon

30.4.4 Schalter und Konfiguration

mini-d install kommt mit einer ganzen Reihe von Schaltern daher. Auch die Einträge für die Konfigurationsdatei sind etwas länger. Als Schalter unterstützt mini-d install die folgenden:

-b (Langform --batch)
  laufe im Stapelmodus (batch mode)

-c (Langform --config=file)
  benutze file als Konfigurationsdatei. Der Standardwert ist ~/.mini-d install.conf

-d (Langform --debug)
  Gib zusätzliche Debuginformationen im Terminal und dem Logfile aus

-k (Langform --kill)
  beende den laufenden Daemon sofort

-n (Langform --no-act)
  führe keine Änderungen aus. Der Schalter ist nützlich in Kombination mit dem Schalter -v

-q (Langform --quiet)
  zeige so wenig Informationen an, wie nur möglich

-r (Langform --run)
  teile dem aktuell laufenden Daemon mit, dass er die Abarbeitungsschlan ge sofort bearbeiten soll

-v (Langform --verbose)
  zeige ausführliche Informationen an

--help
  zeige eine kurze Liste der Schalter an

--no-db
  schalte das Nachschauen in der Paketdatenbank ab. apt-ftparchive läuft ohne den Schalter --db.

--no-log
  schreibe kein Logfile

--version
  zeige die Version von mini-d install an

In der Konfigurationsdatei sind die folgenden Einträge zulässig:

archivedir
  The root of the mini-d install archive. Must be set, either here or on the command line.

extra_keyrings
  Additional GnuPG keyrings to use for signature verification.
incoming_permissions
The permissions for the incoming directory. mini-dinstall will attempt to set the directory’s permissions at startup. A value of zero (’0’ or ’0000’) will disable permission setting. Doing this, you MUST set permission for incoming by hand! Defaults to 0750.

keyrings
GnuPG keyrings to use for signature verification of changes files. Setting this parameter will modify the default list; it is generally better to modify extra_keyrings instead. Defaults to the keyrings from the debian-keyring package.

30.4.5 Lesematerial

• Creating a Package Repository for APT (see https://debian-handbook.info/browse/stable/sect.setup-apt-package-repository.html)

30.5 apt-ftparchive

• Goals: Superset of dpkg-scanpackages and dpkg-scansources
• Pros:
  – Does not rely on any external programs aside from gzip.
  – Creates Release and Contents files without providing *.changes
• Cons:
  – Can be slow on large repositories, unless the input file (?FileList) is sorted first (the sort command works).
• Package: apt
• Distributions: oldstable, stable, testing, unstable, experimental
• Dependencies: unstable/admin/apt-utils
• Automatic repositories: No (Yes with dupload)
• Incoming mechanism: No (Yes with custom move cron script with dupload)
• Pools: Yes
• GPG signing: No (Yes with dupload with script)
• Inclusion of .deb without .changes: ??
• Several versions of each package: ??
• HOWTOs:
  – apt-ftparchive generate Roberto Sanchez how-to — he now recommends to use reprepro
30.6 aptly

- Goals: manage local repositories, snapshot them and publish
- Pros:
  - supports multiple versions of package in one repo
  - has supported for mirroring in the same tool
- Download & Documentation: http://www.aptly.info/
- Distributions: stable, testing, unstable
- Source: GitHub
- Automatic repositories: Yes (?)
- Incoming mechanism: Yes (inoticoming → aptly repo include)
- Pools: Yes
- GPG signing: Yes
- Inclusion of .deb without .changes: Yes
- Several versions of each package: Yes
- HOWTOs:
  - aptly tutorials: https://www.aptly.info/tutorial/
  - Example of setup with incoming: http://oar.imag.fr/wiki/aptly

30.7 debify

- Goal: takes a directory full of Debian packages and creates a signed and properly-structured Debian repository (wrapper around aptly)
- Download: https://hub.docker.com/r/spotify/debify/ (docker container)
- Source: GitHub
Kapitel 31

Einen eigenen APT-Mirror aufsetzen

Wie bereits in „Geeigneten Paketmirror auswählen“ (siehe Abschnitt 3.4) deutlich wurde, stellen die offiziellen Paketmirrors (auch „Spiegelserver“ genannt) stets aktuelle Debian-Pakete für alle Benutzer und Interessierten zur Verfügung. Diese Spiegelserver sind weltweit verteilt und werden weitestgehend individuell betreut.


Der Unterschied zwischen einem Paketcache und einem Paketmirror besteht darin, dass bei ersterem nur die Pakete vorgehalten werden, die bisher nachgefragt wurden, d.h. über die Paketverwaltung bezogen wurden. Im Gegensatz dazu stellt ein Paketmirror die gesamte Distribution inkl. ausgewählter Veröffentlichungen bereit. Er spiegelt den offiziellen, aktuellen Paketbestand und macht ihn somit für die Benutzer, Dienste und Geräte in ihrem 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. An der Stelle ist jedoch dafür zu sorgen, dass der lokale Spiegelserver auch stets aktuelle Pakete vorgehalten. Das gelingt über eine Synchronisation mit einem anderen Paketmirror als Referenz.

Das Debian-Projekt nutzt als Werkzeuge zur Erzeugung und Pflege eines Spiegelserver dak und minidak. dak kürzt Debian Archive Kit ab [DebianArchiveKit], ist die offizielle Lösung für die Spiegelserver auf der Basis von Python und besitzt eine Anbindung an das DBMS PostgreSQL. minidak [minidak] ist eine schlankere Reimplementierung von dak in Form von Shellskripten ohne die Erfordernis einer Datenbank. minidak wird zum Beispiel vom Debian-Ports-Projekt genutzt (siehe dazu Abschnitt 1.2.1). Die offiziellen Empfehlungen für private Spiegelserver, die nicht die gesamte Distribution oder nur eine bestimmte Veröffentlichung bereitstellen, sind reprepro (siehe Abschnitt 31.6) und minid-install (siehe Abschnitt 30.4).

Nachfolgend stellen wir Ihnen mehrere Varianten vor. Diese benutzen die Pakete apt-mirror (siehe Abschnitt 31.1), debmirror (siehe Abschnitt 31.2), debpartial-mirror (siehe Abschnitt 31.3), ubumirror (siehe Abschnitt 31.4), debarchiver (siehe Abschnitt 31.5) und reprepro (siehe Abschnitt 31.6).

Anmerkung
Möchten Sie nach der erfolgreichen Inbetriebnahme ihren eigenen Spiegelserver auch öffentlich zugänglich machen, nutzen Sie dazu am besten das bereits dafür vorbereitete Formular auf der Webseite des Debian-Projekts [Debian-Spiegel-Informationen].

31.1 apt-mirror

• Projektseite [apt-mirror-Projektseite]

• 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]

• Einrichtung anhand eines Beispiels für Ubuntu mit apt-mirror [apt-mirror-ubuntu2]

31.1.1 Wichtige Dateien aus dem Paket

• /etc/apt/mirror.list
• /etc/cron.d/apt-mirror oder crontab
• apt-mirror

31.1.2 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/ bookworm main contrib non-free non-free-firmware

deb http://security.debian.org/ bookworm-security main contrib non-free non-free-firmware

# backports
deb http://ftp.de.debian.org/debian bookworm-backports main contrib non-free non-free-firmware

# 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 :)

31.1.3 Beispiel/HowTo

• nur apt-mirror: [apt-mirror-ubuntu]
• apt-mirror und apt-cacher: [apt-mirror-ubuntu2]

31.1.4 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

31.2 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

```bash
# debmirror -a i386 -s main -s contrib -s non-free
   -h ftp.debian.org
   -d wheezy /home/ftp/debian
   --nosource
   --progress
```

### 31.3 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."

### 31.4 ubumirror

- Paket: [Ubuntu-Paket-ubumirror]

### 31.5 debarchiver

- Goals: Make a simpler version of dak.
- Pros:
  - Easy to use incoming mechanism - even on remote systems - by using a cron-job
  - Packages can be moved into a distribution by
  - reading the Distribution value from .changes file or
  - directly putting the whole package into a distributions-incoming directory.
  - Standard repository (can be pinned)
- Cons:
  - No Pool-architecture at the moment
  - Some useful checks are missing
  - Cleaning needs to be done manually
- Package: debarchiver [Debian-Paket-debarchiver]
- Distributions: oldstable, stable, testing, unstable
- Dependencies: unstable/devel/debarchiver
– adduser, apt-utils (recommended) | dpkg-dev, opalmod (Perl modules), gnupg (optional)

• Automatic repositories: Yes
• Incoming mechanism: Yes
• Pools: No (but suggested somewhere at BTS).
• GPG signing: Yes (with gnupg, post-Sarge feature).
• Inclusion of .deb without .changes: ??
• Several versions of each package: ??

31.6 reprepro

• Paket [Debian-Paket-reprepro]
• Optionen und Filter, um ein Repo zu erzeugen
Kapitel 32

Plattenplatz sparen mit der Paketverwaltung

• Hintergrund:
  – Software besteht aus vielen einzelnen Paketen
  – Speicherplatz reicht nicht immer aus, d.h. ist immer in irgendeiner Form begrenzt
    * Filesystem hat Begrenzungen (Anzahl der Inodes)
    * Partition hat eine physische Grenze
  – je mehr Platz belegt ist, um so langsamer werden die Plattenzugriffe
  – Experimente hinterlassen ihre Spuren in Form von ”Paketresten”, die übriggeblieben sind (aufräumen ist immer nützlich)
• Lösungsmöglichkeiten
  – Hardlink über /usr/share/doc/ laufen lassen [Beckert-Blog-Hardlinking-Duplicate-Files]
  – dpkg --path-exclude=...
  – Paket localepurge, momentan noch unter Kapitel 40
  – Ungenutzte Bibliotheken, denen aber das „Automatisch installiert“-Flag fehlt [Beckert-Blog-Finding-Libraries]
  – den Paketcache aufräumen (siehe Abschnitt 7.5)
  – Auch /var/cache/apt auf tmpfs gehört hier erwähnt, momentan unter Kapitel 29
  – 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.
    * nicht-kritische Pakete entfernen
    * apt umkonfigurieren, so dass es nicht automatisch zusätzliche Pakete installiert
    * Pakete durch kleinere Äquivalente ersetzen
    * unnötige Pakete entfernen
    * unnötige Locales entfernen, bspw. mittels localepurge
    * unnötige Kernelmodule entfernen
  – bleachbit benutzen (https://www.bleachbit.org/)
Kapitel 33

Platte läuft voll

* ja, das gibt es heute immer noch * auch, wenn Festplatten vergleichsweise riesig sind * Hinweis: es gibt ältere Systeme, die klaglos ihren Dienst leisten * also: genügend Anlässe, sich damit auseinanderzusetzen

33.1 Hintergrund

• Pakete werden heruntergeladen, bspw. durch aptitude
• Speicherplatz reicht nicht aus,
  – um alle herunterzuladenden Pakete zu speichern (cachen)
  – um alle benötigten Pakete zu installieren
• wann passiert das
  – ist nicht ungewöhnlich, …
    * wenn System kleiner ausgelegt ist, als gedacht
    * wenn sich das Ziel/der Einsatzzweck der Maschine ändert (Testsystem, Testsystem wird "plötzlich" doch Produktivsystem)
    * neue Paketabhängigkeiten hinzugekommen sind, bspw. für Features, die man vorher nicht brauchte
  – bei der Aktualisierung des Systems (update, upgrade und dist-upgrade)
    * irgendwo müssen die Pakete ja erstmal hin — sprich: also landen die im Paketcache
    * Reihenfolge des Downloads steht nicht wirklich fest
    * Auswahl der Installation . . . zufällig?
33.2 wie löst man diesen Zustand (Empfehlung zum Vorgehen)

- Ziele:
  - produktives, stabiles System
  - alle gewünschten Pakete werden heruntergeladen und installiert
33.3 Varianten

- aptitude verlassen
- mit df -h schauen, wieviel Platz noch verfügbar ist
- mit aptitude clean den Paketcache aufräumen
- mit aptitude autoremove Pakete entfernen, die nicht mehr benötigt werden, aber noch installiert sind
- Paketabhängigkeiten begutachten
  - schauen, ob wir Pakete einzeln (nacheinander) installieren können
  - nach jeder Installation wieder den Paketcache aufräumen

33.4 Fehler beheben

- kurz:
  - geht
- länger:
  - ist etwas unschön
  - ist Kombination aus Automatismen und Handarbeit
  - braucht etwas Zeit und Geduld
- wie bekomme ich heraus, welche Pakete unvollständig installiert sind
  - gibt es eine Bearbeitungs-Queue?
  - wie kann ich mir die anzeigen lassen?
  - kann ich die beeinflussen (was wird zuerst daraus installiert?)
- wie repariere ich die Stück für Stück
  - wie bringe ich dpkg bzw. apt dazu, die Installation zu wiederholen
  - reparieren dpkg/apt das irgendwie von alleine?
  - wie repariere ich Pakete selber, sprich: wie stoße ich eine erneute Installation des Pakets an?
- fehlende Abhängigkeiten selber nachziehen
Kapitel 34

Paketkonfiguration sichern

Betreuen Sie eine Reihe von Rechnern, die jeweils aktuell und zumindest bezogen auf die Paketauswahl identisch bleiben müssen, ist das manuelle Sichern, Pflegen und Einspielen einer Paketliste etwas mühselig. Der händische Abgleich mit Hilfe von dpkg und diff braucht einfach zu lange, ist zudem fehleranfällig und hält Sie von weitaus spannenderen Aufgaben auf ihrem System ab.

Daher schauen wir uns in diesem Kapitel verschiedene Wege an, um diese Schritte möglichst zu automatisieren. Unterteilt haben wir das in das Auslesen der Paketkonfiguration und das Einspielen der zuvor gesicherten Paketkonfiguration. Zum Einsatz kommen dabei die Werkzeuge dpkg (siehe Abschnitt 34.1.1), apt (siehe Abschnitt 34.1.2), debconf-get-selections (siehe Abschnitt 34.1.3), debconf-set-selections (siehe Abschnitt 34.2.2) und apt-clone (siehe Abschnitt 34.1.4 und Abschnitt 34.2.3). Zum Schluß reißen wir eine Auswahl graphischer Werkzeuge kurz an (siehe Abschnitt 34.3).

34.1 Die bestehende Paketkonfiguration auslesen


34.1.1 Mit dpkg

Wie bereits in Abschnitt 8.5 erklärt, ist dpkg das zentrale Werkzeug, um die Paketdatenbank auszulesen und Ihnen die derzeit installierten und vollständig konfigurierten Pakete auf der Standardausgabe (stdout) aufzulisten. Über das nachfolgend gezeigte Kommando erhalten Sie eine entsprechende Liste mit den Namen der Pakete.


Liste der installierten Pakete namentlich sortiert in einer Datei ablegen (Version 1)

```
$ dpkg -l | egrep "^ii" | awk '{ print $2 }' | sort > paketliste
```

Um den Aufwand zu verringern, ist der Schalter --get-selections spannend. Dieser sorgt dafür, daß dpkg nur den Paketnamen samt Installationsstatus ausgibt. Kombiniert mit grep und awk bauen Sie sich daraus eine sortierte Liste wie folgt:

Liste der installierten Pakete namentlich sortiert in einer Datei ablegen (Version 2)
Diese soeben erzeugten Liste nutzen Sie als Referenz, um nun andere Systeme identisch aufzusetzen. Wie das geht, lesen Sie in Abschnitt 34.2.1 nach.

34.1.2 Mit apt

Die Kombination aus apt, awk und tail liefert Ihnen ebenfalls eine Liste der installierten Pakete in alphabetisch aufsteigender Sortierung:

**apt zur Erstellung der Paketliste benutzen**

```bash
# apt list --installed | awk -F "/" '{ print $1 } ' | tail -n +2 > paketliste
```

Bitte beachten Sie, daß apt nur die Namen der Pakete ermittelt. Sämtliche Konfiguration bleibt unbeachtet.

34.1.3 Mit debconf-get-selections


Die Ausgabe von debconf-get-selections umfaßt den Paketnamen und dessen Konfiguration, sprich: die Fragen plus die Antworten, aus denen Sie bei der Installation des Paketes durch APT wählen können sowie ihre tatsächliche Antwort. Der nachfolgende Ausschnitt zeigt das für das Paket `cups`:

**Die Ausgabe von debconf-get-selections**

```bash
...# Backends für die Kommunikation mit dem Drucker:
# Choices: lpd, Socket, USB, SNMP, dnssd
cups     cupsys/backend multiselect  lpd, socket, usb, snmp, dnssd
...```

Abbildung 34.1 zeigt den dazugehörigen Konfigurationsdialog, den Sie auch stets mit Hilfe von dpkg-reconfigure cups erreichen können.
Abbildung 34.1: Fragen zum Paket cups

Die Ausgabe von `debconf-get-selections` erfolgt auf der Standardausgabe (`stdout`). Eine Liste in einer Datei `paketliste`, die lediglich die Paketnamen in sortierter Reihenfolge enthält, erzeugen Sie mit Hilfe der Kombination aus den fünf Werkzeugen `debconf-get-selections`, `grep`, `awk`, `sort` und `uniq` wie folgt:

**Auslesen der bestehenden Paketkonfiguration aus der Debconf-Datenbank**

```bash
$ debconf-get-selections | grep -v "^#" | awk '{ print $1 }' | sort | uniq > paketliste
```

In Abschnitt 34.2.2 lesen Sie, wie Sie die erzeugte Liste benutzen, um die darin genannten Pakete auf einem anderen System wieder einzuspielen.

### 34.1.4 Mit `apt-clone`

Dieses Werkzeug steht über die gleichnamigen Pakete bei Debian und Ubuntu bereit (siehe [Debian-Paket-apt-clone] und [Ubuntu-Paket-apt-clone]). Es sieht sich selbst als *helper script*, welches das Klonen einer bestehenden Linuxinstallation vereinfacht.


Es stellt beim Aufruf nach Möglichkeit auf dem Bestandssystem so viele Informationen zur zu klonenden Installation zusammen, wie möglich sind. Das umfasst:

- die Paketquellen über die Datei `/etc/apt/sources.list`
- die Paketquellen über das komplette Verzeichnis `/etc/apt/sources.list.d` samt Inhalt
- die hinterlegten Voreinstellungen über das Verzeichnis `/etc/apt/preferences.d/
- den Debian-Schlüsselring mit den darin hinterlegten GPG-Schlüsseln für die verwendeten Paketquellen unter `/etc/apt/trusted.gpg` und dem Verzeichnis `/etc/apt/trusted.gpg.d`
- den Paketstatus aus der Paketdatenbank
• die Pakete, die nicht mehr von den Paketquellen heruntergeladen werden können. Diese landen ebenfalls unter /var/lib/apt-clone im Archiv. Dazu bedient sich apt-clone des Werkzeugs dpkg-repack [Debian-Paket-dpkg-repack]. Im nachfolgenden Beispiel sehen Sie, dass das nicht immer sauber gelingt und auch zu Paketen führen kann, die defekt sind (broken) — d.h. unvollständige Abhängigkeiten besitzen — und nicht ohne weiteres wiedereinspielbar sind.

Bevor Sie apt-clone clone ausführen, legen Sie ein (lokales) Verzeichnis fest, in dem das Archiv abgelegt werden soll. Im nachfolgenden Beispiel heißt das lokale Verzeichnis schlicht und einfach packagelist. Ergebnis des Aufrufs von apt-clone ist ein Archiv im Format tar.gz, welches Sie auf dem Zielsystem wieder mit Hilfe von apt-clone einspielen (siehe Abschnitt 34.2.3).


apt-clone sammelt Informationen

```bash
# apt-clone clone --with-dpkg-repack packagelist/.

dpkg-deb: building package 'sge' in './sge_8.1.8_amd64.deb'.

dpkg-deb: building package 'libnccl2' in './libnccl2_2.3.5-2+cuda10.0_amd64.deb'.

dpkg-repack: warning: unknown information field 'Original-Maintainer' in input data in → entry in dpkg's status file

dpkg-deb: building package 'lesstif2' in './lesstif2_0.95.2-1_amd64.deb'.

dpkg-repack: warning: unknown information field 'Original-Maintainer' in input data in → entry in dpkg's status file

dpkg-deb: building package 'libcudnn7' in './libcudnn7_7.3.1.20-1+cuda10.0_amd64.deb'.

dpkg-repack: warning: unknown information field 'Original-Maintainer' in input data in → entry in dpkg's status file

dpkg-deb: building package 'libcudnn7-dev' in './libcudnn7-dev_7.3.1.20-1+cuda10.0_amd64. deb'.

dpkg-deb: error: conffile '/opt/sge/util/install_modules/inst_template.conf' does not ← appear in package


dpkg-repack: Problems were encountered in processing.

dpkg-repack: The package may be broken.

not installable: sge, libnccl2, lesstif2, libdb5.1, libcudnn7, libcudnn7-dev, sge-common, ←

cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48, libnccl-dev, libxp6, db5.1-util, libdb5 ←
       .1++

version mismatch: libssl1.1, mdadm, python3-distutils, libitm1, libmagic-mgc, samba-libs, ←

lxd-client, wget, postfix, cpp,

...#
```


**Informationen zum erzeugten apt-clone-Archiv anzeigen**

```bash
$ apt-clone info apt-clone-state-kiste.tar.gz

Hostname: kiste
Arch: amd64
Distro: bionic
Meta:
Installed: 1301 pkgs (751 automatic)
Date: Tue Oct 15 14:55:03 2019
$
```

Bitte beachten Sie bei der Verwendung von apt-clone noch die folgenden Punkte:

• Das Zielsystem muss das gleiche Betriebssystem und die gleiche Veröffentlichung wie das Originalsystem besitzen.

• apt-clone gleicht den Paketbestand des Originalsystems mit den Paketquellen ab. Es merkt an, wenn installierte Pakete nicht mehr aktuell sind und im Paketmirror bereits eine neuere Version vorliegt. Um das o.g. erzeugte Archiv möglichst klein zu halten, aktualisieren Sie das Originalsystem vor dem Aufruf von apt-clone, sofern das möglich und auch praktikabel ist (siehe Abschnitt 8.41) und dem Vorgehen nichts entgegen spricht.
• Räumen Sie ihr Originalsystem vor dem Klonen auf. Entfernen Sie nicht mehr benötigte Software und auch verwaiste Pakete, bspw. über das Kommando `apt-get autoremove` (siehe Abschnitt 8.44).

• `apt-clone` benachrichtigt Sie, wenn Dateien vorliegen, die nicht im Originalpaket enthalten sind, bspw. Konfigurationsdateien. Es ist so angelegt, dass es bestehende, geänderte Dateien übernimmt und somit eine exakte Kopie des Zustands des installierten Pakets erzeugt.

• `apt-clone` gibt am Ende eine Liste der Pakete aus, die es für nicht installierbar hält.

• Führen Sie `apt-clone` später auf dem Zielsystem aus, überschreibt es ihre bereits bestehende Paketliste. Es löscht Pakete bzw. installiert fehlende nach (siehe Abschnitt 34.2.3).

### 34.2 Eine gesicherte Paketkonfiguration wieder einspielen

#### 34.2.1 Mit `apt-get`

Haben Sie eine Paketliste wie in Abschnitt 34.1.1 erzeugt, ist das Einspielen dieser Liste auf einem neuen System vergleichsweise einfach, bspw. mit diesem Aufruf:

**Einspielen der gespeicherten Paketliste mit Hilfe von `apt-get`**

```bash
# apt-get install < paketliste
... 
#
```

Bitte beachten Sie, dass in der Paketliste keine Konfigurationsdateien enthalten sind. Beim Einspielen bzw. Installieren des jeweiligen Pakets werden die Dateien daraus extrahiert oder neu erzeugt.

#### 34.2.2 Mit `debconf-set-selections`


- `-c` *(Langform `--checkonly`)*
  Eingabedatei nur auf Fehler prüfen

- `-v` *(Langform `--verbose`)*
  ausführliche Ausgabe beim Einspielen

Über den folgenden Aufruf spielen Sie die gespeicherte Konfiguration als Benutzer mit administrativen Rechten wieder ein:

**Einspielen der gespeicherten Konfiguration mit Hilfe von `debconf-set-selections`**

```bash
# debconf-set-selections paketliste
... 
#
```

Verwenden Sie eine Datei, die lediglich aus den Namen der Pakete besteht, hilft Ihnen dieses Kommando beim Wiedereinspielen:

**Einspielen der gespeicherten Paketliste mit Hilfe von `xargs` und `apt-get`**

```bash
# xargs -a "paketliste" apt-get install -y
... 
#
```
34.2.3 Mit apt-clone

Haben Sie zuvor ein Archiv wie unter Abschnitt 34.1.4 beschrieben erstellt, lernen Sie nun, wie Sie das auf dem Zielsystem einspielen. Als erstes übertragen Sie das Archiv auf ihr Zielsystem, bspw. per USB-Stick, externe Festplatte oder mit Hilfe des Kommandos `scp`.

Ist das erfolgt, rufen Sie auf dem Zielsystem `apt-clone` als administrativer Benutzer mit dem Schalter `restore` und dem Namen des zuvor erzeugten Archivs auf. Daraufhin entpackt `apt-clone` das Archiv und spielt die darin enthaltenen Pakete samt deren Konfiguration auf wieder dem Zielsystem ein.

### Mit apt-clone erzeugte Paketkonfiguration wieder einspielen

```bash
# apt-clone restore apt-clone-state-kiste.tar.gz
...
#
```


Das Werkzeug `apt-clone` bietet zwei Schalter an:

- `restore` packe das Archiv aus und spiele den Paketbestand auf dem Zielsystem ein
- `restore-new-distro` packe das Archiv aus, spiele den Paketbestand auf dem Zielsystem ein und aktualisiere diesen, sofern möglich

Letzteres kann genutzt werden, um das Einspielen und Aktualisieren eines Systems in einem einzigen Aufruf durchzuführen.

34.3 Graphische Werkzeuge

34.3.1 Aptik


34.3.2 Mintbackup

Für Linux Mint steht das Werkzeug `mintbackup` bereit [mintbackup]. Es ist ein graphisches Werkzeug, welches Paketlisten sichern und wieder einspielen kann.
Abbildung 34.2: Sicherungsdialog von Mintbackup
Kapitel 35

Automatisierte Installation

35.1 Einstieg


Drei Begriffe, die häufig in diesem Zusammenhang genutzt werden, sind:

*Hardware provisioning und OS provisioning* das vorherige Auswählen von Softwarekomponenten für die im System verbaute Hardware und das darauf genutzte Betriebssystem, bspw. der Linux-Kernel und das Paket für den passenden Grafiktreiber samt Einstellungen.

*Konfigurationsmanagement* das Verwalten und Pflegen der zuvor definierten Rechnerklassen samt Liste der Softwarepakete und den Einstellungen zum jeweiligen Paket.

Unabhängig davon, wieviele Einzelrechner oder Rechenknoten letztendlich im Verbund zu berücksichtigen sind, gibt es für jede Rechnerklasse stets entweder eine bestehende Referenzinstallation (genannt Master) oder zumindest eine hinterlegte Konfiguration, auf deren Basis sich die weiteren Rechner identisch zur Referenz aufsetzen lassen. Dieses Kapitel beleuchtet anhand verschiedener Vorgehensweisen, wie Sie diesen Schritt in der Praxis umsetzen. Übliche Varianten sind:


35.2 Kriterien für die Auswahl einer geeigneten Lösung

Folgende Punkte halten wir bei der Auswahl einer geeigneten Lösung für hilfreich. Diese Punkte haben wir dem Debian-Wiki entnommen (siehe dazu [Debian-Wiki-Automated-Installation]):

unbeaufsichtigtes Ablaufen
   Installationen sollten ohne oder mit möglichst wenig menschlicher Interaktion ablaufen

parallele Installation
   mehrere Systeme werden gleichzeitig installiert

Skalierbarkeit
   neue Klienten und Veröffentlichungen von Betriebssystemen können einfach integriert werden

Flexibilität
   eine Konfiguration sollte möglich sein, die viele Eigenschaften erlaubt

Anpassbarkeit
   es sollte möglich sein, das System nach Bedarf an spezifische Umgebungen und Gegebenheiten anzupassen

Bitte passen Sie diese Liste an ihre jeweiligen Gegebenheiten an.

35.3 Mit Debian-Werkzeugen

- Rückgriff auf Beschreibung aus Paketkonfiguration sichern in Kapitel 34
- Preseeding (siehe https://fak3r.com/2011/08/18/howto-automate-debian-installs-with-preseed/)
- dpkg
- debconf-get-selections
- debconf-get-selections
- apt-clone

35.4 Komplette Lösungen

35.4.1 Cobbler

- https://cobbler.github.io/

35.4.2 FAI

- Fully-automatic installation (FAI) [FAI-Projekt]
- Eintrag im Debian Wiki [Debian-Wiki-FAI]
- Beitrag im Linux Magazin 01/09 [FAI-Bornemann-Karg]
- FAI 5.2 bringt Cloud-Support [FAI-Cloud-Support]
35.4.3 Kickstart

- Textdatei, welche die Systemkonfiguration und das Setup beschreibt
  - Disklayout und Dateisysteme
  - Bootloader
  - Netzwerkkonfiguration
  - Benutzer und Zugänge
  - Softwareeinrichtung
- nicht für Debian verfügbar
- verfügbar für Ubuntu (siehe https://doc.ubuntu-fr.org/kickstart)
  - graphisches Werkzeug im Paket system-config-kickstart [Ubuntu-Paket-system-config-kickstart]

35.5 Erfahrungen aus der Praxis

ToDo
Kapitel 36

Automatisierte Aktualisierung

• Ziel:
  – eine ganze Menge von identisch aufgesetzten Rechnern aktualisieren, bspw. einen Schulungsraum oder Cluster

36.1 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 37

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.


37.1 Nicht installierte Pakete mit lintian prüfen

37.1.1 lintian verstehen


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 37.1) und
- automatisierte Ablehnung von frisch hochgeladenen Paketen bei groben Fehlern\(^1\).

\(^1\) Von lintian bemerkte, besonders schwere Fehler sollten bei offiziellen Paketen gar nicht auftauchen, da diese damit sozusagen bereits beim Aufnahme-test durchfallen.
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 37.1 zusammengestellt haben.

Tabelle 37.1: Klassifikation der lintian-Fehlermeldungen

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Beschreibung</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Fehler (error)</td>
</tr>
<tr>
<td>W</td>
<td>Warnungen (warning)</td>
</tr>
<tr>
<td>I</td>
<td>Informationelle Hinweise (informational tags)</td>
</tr>
<tr>
<td>P</td>
<td>Pingelige Markierung (pedantic tags)</td>
</tr>
<tr>
<td>O</td>
<td>Überschriebene Markierungen (overridden tags)</td>
</tr>
<tr>
<td>X</td>
<td>Experimentelle, ggf. fehleranfällige Markierungen (experimental tags)</td>
</tr>
<tr>
<td>N</td>
<td>Kein Fehler, Bemerkung (note)</td>
</tr>
</tbody>
</table>

37.1.2 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 37.2 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:

\[-v\, (\text{Langform } --\text{verbose})\]

für eine ausführlichere Ausgabe (verbose).

\[-c\, (\text{Langform } --\text{color})\]

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).

\[-i\, (\text{Langform } --\text{display-info})\]

Auflistung der informationellen Markierungen (info tags). So bleiben auch Schreibfehler nicht unentdeckt.

\[-x\, (\text{Langform } --\text{display-experimental})\]

Auflistung der experimentellen Markierungen.

\[-p\, (\text{Langform } --\text{pedantic})\]

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
Die Ausgabe auf der Kommandozeile und in Abbildung 37.1 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 37.1: 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: 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: Refer to Debian Policy Manual section 5.6.23 (Homepage) for details.
N: Severity: pedantic, Certainty: possible
N: Check: fields, Type: binary, udeb, source
N: W: mp4h: postinst-has-useless-call-to-ldconfig
N: The postinst script calls ldconfig even though no shared libraries are
N: installed in a directory controlled by the dynamic library loader.
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: Refer to Debian Policy Manual section 8.1.1 (ldconfig) and
N: Severity: minor, Certainty: certain
N: Check: shared-libs, Type: binary, udeb
[...]
```

37.2 Bereits installierte Pakete mit adequate prüfen

Im Gegensatz zu lintian (siehe Abschnitt 37.1) validieren Sie mit adequate [Debian-Paket-adequate] die Pakete, die bereits auf ihrem System installiert sind. adequate steht Ihnen ab Debian 8 Jessie 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 Tag2. Alle angegebenen Tags trennen Sie durch Komma in einer Liste.

- **--no-tags Tag1,Tag2**
  - Beschränkung der Ausgabe auf die angegebenen Fehlerwerte ohne Tag1 und Tag2. Alle angegebenen Tags trennen Sie durch Komma in einer Liste.

Als Tags sind zulässig:

- `bin-or-sbin-binary-requires-usr-lib-library`
• broken-binfmt-detector
• broken-binfmt-interpreter
• broken-symlink
• incompatible-licenses
• library-not-found
• missing-alternative
• missing-copyright-file
• missing-symbol-version-information
• obsolete-conffile
• program-name-collision
• py-file-not-bytecompiled
• pyshared-file-not-bytecompiled
• symbol-size-mismatch
• undefined-symbol

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
```


Überprüfung des gesamten Paketbestands (Ausschnitt)

```
$ adequate --all
unknown-horizons: broken-symlink /usr/share/unknown-horizons/content/fonts/Unifont.ttf -> ../../../fonts/truetype/unifont/unifont.ttf
moka-icon-theme: broken-symlink /usr/share/icons/Moka/16x16/apps/org.gnome.Builder.png -> builder.png
moka-icon-theme: broken-symlink /usr/share/icons/Moka/16x16/apps/org.gnome.Maps.png -> gnome-maps.png
...
```

37.3 Bugreports anzeigen

37.3.1 Hintergrundwissen

Allgemein gesprochen, ist ein Bugreport ein Fehlerbericht zu einem aufgetretenen bzw. von Ihnen bemernten 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 Qualitätsvor-
gaben. 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 37.3.2) aus und warnt Sie, falls für das Paket kritische Fehler im Debian BTS hinterlegt
sind. Die Entscheidung liegt dann bereits 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 oben bereits 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 Fehlerbe-
richts, 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 37.1), `apt-listbugs` (siehe Abschnitt 37.3.2) und `popbugs` (siehe Abschnitt 37.3.4) zur Verfügung.

Finden Sie einen Fehler bezüglich eines Debianpakets, schreiben Sie am besten einen Fehlerbericht (bug report). Eine ausführli-
che Beschreibung dessen, auf welche Punkte Sie bei dem Fehlerbericht Wert legen 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].

37.3.2 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 37.3.1 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):

- `--severity` (Langform `--severity`)
  Fehler je nach Schweregrad eingrenzen. Möglich sind die Werte critical (kritisch), grave (sehr schwerwiegend),
  serious (schwerwiegend), important (wichtig), normal (normal), minor (weniger relevant), wishlist (Wunsch-
  liste) und all (alle Schweregrade). Mehrere Werte trennen Sie mittels Komma voneinander. Der Standardwert ist die
  Kombination der drei erstgenannten Werte critical, grave, serious.

- `--tags` (Langform `--tags`)
  Filtre die Fehlermeldungen anhand des gegebenen Schlüsselworts. Mehrere Werte trennen Sie mittels Komma voneinan-
  der.
Debian-Paketmanagement

-S **Status (Langform --stats)**
Filtern und sortieren Sie die Fehlerberichte anhand des aktuellen Status. Mögliche Statuswerte sind pending (offener Fehler), forwarded (Fehlerbericht ist als weitergeleitet markiert), pending-fixed (Fehlerbericht ist als gelöst markiert, aber noch ohne Bestätigung), fixed (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)**
Filtern Sie die Fehlerberichte anhand der gegebenen Nummer des Fehlerberichts und zeige nur die betreffenden an. Mehrere Werte trennen Sie mittels Kommas voneinander.

-D **(Langform --show-downgrade)**
Zeigen Sie nur die Fehlerberichte für Pakete an, für die ein Downgrade erfolgt ist (siehe auch Abschnitt 8.42).

-P **Priorität (Langform --pin-priority)**
Benutzen Sie die Pin-Priorität als Filterkriterium (siehe Kapitel 20 für weitere Informationen zur Pin-Priorität).


Berichte für das Paket `coreutils` auflisten:

```bash
$ 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)
```

37.3.3 **Ergänzende Bugreports mit apt-listchanges herausfiltern**


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.

--which=(news|changelogs|both): Wählt aus, welche Änderungen angezeigt werden. news schränkt auf NEWS.Debian ein, changelogs hingegen auf changelog.Debian. both wählt beide Informationsquellen aus.

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
```

37.3.4 Release-kritische Fehler mit popbugs finden


```bash
$ popbugs -o fehlerliste.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 37.2: Auflistung der bekannten, kritischen Fehler nach der Analyse von popbugs

37.3.5 Release-kritische Fehler mit rc-alert finden


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)

```plaintext
$ rc-alert apt
Package: apt
Bug: 558784
Title: apt: re-adds removed keys
Flags: [ ] (lenny-ignore or squeeze-ignore)
Dists: [S] (stable)

... Package: apt Bug: 776910
```
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 oder experimental. Damit erhalten Sie zunächst alle Fehler, die entweder in den Veröffentlichungen 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 \(--include-dist-op\) 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)
```

```
# 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
```

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
```
Für eine weitere Recherche mit ausführlicher Anleitung zu `rc-alert` empfehlen wir Ihnen den Beitrag von Gambaru [gambaru-rc-alert].

### 37.3.6 Welche der von mir genutzten Pakete benötigen Hilfe?


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.


- `-a` (Langform `--all`)
  Möglichkeiten für alle Pakete anzeigen, nicht nur für die installierten Pakete.

- `-o` (Langform `--old`)
  Zeige nicht nur die neu hinzugekommernen 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

testing-autorm
Kennzeichnung für Pakete, die in Kürze aus dem Zweig Debian testing entfernt werden. Hintergrund sind zumeist bisher nicht behobene, veröffentlichtskritische 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, veröffentlichtskritischer 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.


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
  ...
```

37.4 Auslaufende Sicherheitsaktualisierungen mit check-support-status anzeigen


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 Sicherheitsaktualisierungen lediglich eingeschränkt wird.
Abbildung 37.3: 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
```

**--type Variante**
bezeichnet die Art der Einschränkung der Sicherheitsaktualisierungen. Zur Auswahl stehen ended und limited für zukünftig endende bzw. eingeschränkte Sicherheitsaktualisierungen.
Kapitel 38

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 39

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 zusammenge stellt, ausführlich getestet und — falls alles gutgeht und freigegeben wurde — auf dem Produktivsystem ausgerollt.


- 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 ←
```

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 40

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.

40.1 Batterielaufzeit

Todo

40.2 Speicherplatz

Bekannte Pakete:

- docpurge [docpurge]
- eatmydata [Debian-Paket-eatmydata]: verringert die Menge der Synchronisationen und somit die Schreibzugriffe

40.3 localepurge


Nach der ersten Benutzung wird localepurge automatisch nach jedem Aufruf 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
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 8.32).
- 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 8.39).

40.4 Bildschirmgröße

Todo

40.5 Erfordernisse an die CPU

- Debdelta // gehört hier erwähnt, bzw. systeme-mit-schlechter-anbindung-warten.adoc
- dietlibc über das Paket dietlibc-dev [Debian-Paket-dietlibc-dev]
Kapitel 41

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 Lösungswege sind vorausschauendes Denken und Handeln von Vorteil.

41.1 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.


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.

41.2 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 den Werkzeugen und Verfahren, die wir bislang im Buch bereits angesprochen haben. Nachfolgend beleuchten wir die Verfahren noch etwas genauer.

Sagen Ihnen unsere Empfehlungen nicht zu, werfen Sie bitte einen genaueren Blick auf die drei Projekte bzw. Werkzeuge apt-offline, dpkg-split und Keryx. Diese stellen wir Ihnen im Anschluss in Abschnitt 41.3 genauer vor.

### 41.2.1 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 8.34 und Abschnitt 8.35. Das betrifft nicht nur APT, sondern auch aptitude.

### 41.2.2 Einbindung fester Installationsmedien


**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.


### 41.2.3 Einbindung eines lokalen Paketmirrors

Variante Drei ist das Benutzen eines eigenen Paketmirrors. Dieser kann lokal vorliegen (siehe Kapitel 31), 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 ihrer Paketquellen wie folgt aus:

**Einbinden einer externen Festplatte als Paketmirror**

```
deb file:/mnt/mirror/debian stable main contrib non-free
```


41.3 Werkzeuge

41.3.1 Offline-Verwaltung mit apt-get und wget


awk ist ein Analyse- und Filterprogramm für Textdaten [Debian-Paket-gawk]. Mit wget beziehen Sie hingegen Dateien über das Netzwerk [Debian-Paket-wget] auf der Basis von HTTP, HTTPS und FTP. Während awk zu den essentiellen Programmen zählt, ist wget optional und daher ihrerseits vor dessen Verwendung noch zu installieren, falls es noch nicht auf ihrem System vorhanden ist.

Benötigt wird ein Rechner ohne Internetzugang (genannt rechner-offline) und ein Rechner mit Internetzugang (nachfolgend rechner-online genannt). Beide Rechner verfügen über den gleichen Paketbestand.

Auf rechner-online ist im Verzeichnis /usbplatte ein externer, mobiler Datenträger eingebunden, um darauf die Anweisungen zum nachfolgenden Bezug der Pakete und die Pakete selbst zu speichern. Diese spielen Sie dann auf rechner-offline ein. Der Ablauf zum Bezug der Pakete umfasst zunächst die nachfolgenden Schritte und beginnt auf rechner-online:

1. Aktualisieren der lokalen Paketdatenbank mittels apt-get update oder apt update
2. Erzeugen der URLs zum Bezug der aktualisierten Pakete mittels apt-get. Umleitung der Ausgabe von apt-get in die lokale Datei uris

Hier sehen Sie alle Anweisungen auf der Basis von apt-get. Der Schalter -qq von apt-get ist die verschärfte Form von -q (Langform: --quiet) und sorgt dafür, dass apt-get eine „vollkommen stille“ Ausgabe erzeugt, die für unsere Automatisierung geeignet ist. Der Schalter --print-uris gibt die URLs aus, mit deren Hilfe wget die Pakete herunterladen kann.

Aufrufreihenfolge mittels apt-get, awk und wget

```
# apt-get update
# apt-get -qq --print-uris upgrade > uris
# awk '{print "wget -O " $2 " " $1\}' < uris > /usbplatte/wget-script
# cd /usbplatte
# sh -x ./wget-script
```

Die Datei uris enthält Einträge in folgender Form, hier beispielhaft für ImageMagick und GCC 4.9, die von einem Freexian-Mirror für eine LTS-Version bezogen werden. In der ersten Spalte befindet sich die URL, danach folgt der Name der Paketdatei und die MD5-Summe zum Paket.

URLs von bezogenen Paketen
Das Skript für `wget` enthält dann Aufrufe in folgender Form:

**Pakete per wget beziehen**

```
wget -O libmagickcore-6.q16-2_8%3a6.8.9.9-5+deb8u26_amd64.deb 'http://deb.freexian.com/extended-lts/pool/main/i/imagemagick/libmagickcore-6.q16-2-extra_6.8.9.9-5+deb8u26_amd64.deb'
wget -O linux-compiler-gcc-4.9-x86_3.16.84-1_amd64.deb 'http://deb.freexian.com/extended-lts/pool/main/l/linux/linux-compiler-gcc-4.9-x86_3.16.84-1_amd64.deb'
```

Rufen Sie das Skript auf, holt `wget` ein Paket nach dem anderen von der angegebenen URL. Als Ergebnis dieser Aufrufe finden Sie auf dem mobilen Datenträger alle `deb`-Pakete, die zur Aktualisierung mittels `apt-get upgrade` erforderlich sind. Hängen Sie nun den mobilen Datenträger auf `rechner-online` aus und binden Sie diesen nun auf `rechner-offline` ein.


Eine Aktualisierung mit einem externen Paketchache

```
# apt-get -o dir::cache::archives="/usbplatte/*" upgrade
```

Ab jetzt verfügt der Rechner ohne Internetzugang wieder über einen aktuellen Paketbestand.

### 41.3.2 Das Projekt apt-offline

#### 41.3.2.1 Projektidee


#### 41.3.2.2 Benutzung im Detail

In Schritt 1 erzeugen Sie eine Signatur für die Maschine, die aktualisierte Paketinformationen und die Pakete bekommen soll. Das erfolgt auf der Maschine, die offline ist. Dazu benutzen Sie das nachfolgende Kommando mittelt `apt-offline`. Der verwendete Schalter `set` erzeugt die Signaturdatei, welchen Sie im Aufruf nach dem Schalter benennen. Bezüglich des Dateinamens gibt keine Vorgabe — hier haben wir beispielhaft die Datei `/tmp/apt-offline.sig` gewählt.

```
# apt-offline set /tmp/apt-offline.sig
```

Alternativ kennt `apt-offline` noch die folgenden Schalter für andere Aktionen:
Debian-Paketmanagement

--bug-reports
hole auch die Bugreports für die Pakete

--bundle
erzeuge ein Archiv mit allen benötigten Paketen

get
beziehe die Paketinformationen

--install-packages
installiere das Paket

--threads
Anzahl paralleler Threads, die bei der Ausführung benutzt werden sollen

--update
nur Updates (Einspielen aktualisierter, fehlerbereinigter Pakete mit der gleichen Versionsnummer)

--upgrade
nur Upgrades (Einspielen aktualisierter, fehlerbereinigter Pakete mit einer neueren Version)

ohne Optionen
alles auf den allerneuesten Stand bringen (entspricht einem dist-upgrade)

Das folgende Beispiel zeigt, wie Sie die Paketinformationen für die offline-Maschine holen, die auch Bugreports enthält. Die Ausführung erfolgt über 5 parallele Threads. Diese Kommando führen Sie als Schritt 2 noch auf der Maschine aus, die nicht über eine Netzanbindung verfügt.

```
# apt-offline get apt-offline.sig --bug-reports --threads 5
```

Ergebnis obigen Aufrufs sind Zusatzinformationen in der Datei apt-offline.sig, die Sie nun auf einen Datenträger kopieren.

Für ein einzelnes Paket sieht der Aufruf etwas anders aus. Um die Installation des Pakets approx [Debian-Paket-approx] vorzubereiten, benutzen Sie diesen Aufruf:

```
# apt-offline set --install-packages approx apt-offline.sig
```

Nun beziehen Sie die gewünschten Pakete. Als Schritt 3 führen Sie den nachfolgenden Aufruf auf dem Rechner mit Netzanbindung aus:

```
# apt-offline get --bundle bundle.zip apt-offline.sig
```


```
# apt-offline install /media/USB/bundle.zip
```

Die Pakete landen im lokalen Paketcache und wirken für APT jetzt so, als wenn sie vorher schonmal bezogen wurden. Mittels apt-get install Paketname spielen Sie jetzt das gewünschte Paket ein.

### 41.3.3 Pakete mit dpkg-split aufteilen


dpkg-split kommt mit einer ganzen Reihe von nützlichen Schaltern:
-? (Langform: --help)
Zeigt die Hilfeseite zu dpkg-split an.

-a (Langform: --auto)
Reiht die Stücke automatisch in die Warteschlange und setzt, falls möglich, das Paket wieder zusammen.

-d (Langform: --discard)
Löscht Einträge aus der Warteschlange derer, die auf die verbleibenden Stücke ihrer Pakete warten.

-I (Langform: --info)
Gibt Informationen zu einem Stück aus, welches zuvor von dpkg-split erzeugt wurde.

-j (Langform: --join)

-l (Langform: --listq)
Listet den Inhalt der Warteschlange der wieder zusammenzubauenden Pakete auf.

-o (Langform: --output)

-Q (Langform: --npquiet)
Führen Sie ein automatisches Einreihen oder Wiederzusammensetzen durch, gibt dpkg-split normalerweise eine Mel- dung aus, falls ein übergebener Teil kein Binärpaketteil ist. Dieser Schalter unterdrückt diese Meldung, um Programmen wie dpkg zu erlauben, sowohl mit geteilten als auch ungeteilten Paketen umzugehen, ohne störende Meldungen zu erzeugen.

-S (Langform: --partsize)
Legt die Größe der Stücke fest. Die Angabe erfolgt in Kilobytes zur Basis 1024 Byte. Die Standardgröße beträgt 450 KB.

-s (Langform: --split)
Teilt ein einzelnes Debian-Binärpaket in mehrere Stücke. Die Benennung der Ausgabedateien folgt Paketname.nofm.deb mit n als Stücknummer und m als Anzahl der Stücke.

Geben Sie im Aufruf kein Präfix an, wird der Dateiname vom Komplettarchiv entnommen, einschließlich Verzeichnis, wobei das abschließende .deb entfernt wird.

--depotdir
Spezifiziert ein alternatives Verzeichnis für die Warteschlange von den Stücken, die auf automatisches Wiederzusammen- führen warten. Standardmäßig ist dies das Verzeichnis /var/lib/dpkg.

--msdos
Erzwingt, dass die von -s erzeugten Ausgabedateinamen MSDOS-kompatibel sind.

Dies verstümmelt das Präfix — entweder den voreingestellten aus dem Eingabedateinamen abgeleiteten oder den als Argument übergebenen. Dabei werden alphanumerische Zeichen durch Kleinbuchstaben und Pluszeichen durch x ersetzt sowie alle anderen Zeichen entfernt. Das Ergebnis wird dann soweit wie nötig abgeschnitten. Ergebnis ist ein Dateiname der Form Prädixnofm.deb mit n als Stücknummer und m als Anzahl der Stücke.

--version
Gibt die verwendete Version von dpkg-split aus.

Nachfolgend erklären wir anhand des Paketes xsnow, wie Sie dpkg-split benutzen.

Beispiel 1 ist das Zerlegen des Pakets xsnow in Stücke mit einer Größe von 10 KB. Zum Einsatz kommen im Aufruf die beiden Schalter -s (zerlegen) und -S 10. Letzteres bestimmt die Größe der Ausgabedatei von maximal 10 KB.

Als Ergebnis erhalten Sie vier Stücke. Diese werden in der Form Paketname.nofm.deb mit n als Stücknummer (im Beispiel die Werte 1 bis 4) und m als Anzahl der Stücke (im Beispiel 4) benannt.

Zerlegen des Debianpakets xsnow in Stücke zu 10 KB
$ dpkg-split -s -S 10 xsnow_1%3a1.42-9_amd64.deb
Paket xsnow wird in 4 Teile aufgeteilt: 1 2 3 4 fertig
$ ls -lh
insgesamt 136K
-rw-r--r-- 1 frank frank 9,2K Jan 7 11:09 xsnow_1%3a1.42-9_amd64.1of4.deb
-rw-r--r-- 1 frank frank 9,2K Jan 7 11:09 xsnow_1%3a1.42-9_amd64.2of4.deb
-rw-r--r-- 1 frank frank 9,2K Jan 7 11:09 xsnow_1%3a1.42-9_amd64.3of4.deb
-rw-r--r-- 1 frank frank 8,2K Jan 7 11:09 xsnow_1%3a1.42-9_amd64.4of4.deb
$

In Beispiel 2 zeigen Sie die Informationen zu einem Stück eines mittels dpkg-split zerlegten deb-Pakets an. Dazu benutzen Sie den Schalter \(-I\) gefolgt von einem Dateiname als Parameter, hier: xsnow_1%3a1.42-9_amd64.3of4.deb. Die Ausgabe umfasst die Formatversion, die Paketinformationen, die Nummer des Stücks (hier 3 von 4), die Größe ("Teil-Länge") sowie den Offset im Originalpaket ("Teil-Offset").

Informationen zum Teil xsnow_1%3a1.42-9_amd64.3of4.deb erhalten

$ dpkg-split -I xsnow_1%3a1.42-9_amd64.3of4.deb
xsnow_1%3a1.42-9_amd64.3of4.deb:
  Teil-Formatversion: 2.1
  Teil des Paketes: xsnow
    ... Version: 1:1.42-9
    ... Architektur: amd64
    ... MD5-Prüfsumme: 3ddeabaec77416662e45de36d9960a2a
    ... Länge: 35836 Byte
    ... geteilt alle: 9216 Byte
  Teil-Nummer: 3/4
  Teil-Länge: 9216 Byte
  Teil-Offset: 18432 Byte
  Teil-Dateigröße (ben. Anteil): 9418 Byte
$

Das Beispiel 3 zeigt, wie Sie ein Paket wieder aus einzelnen Stücken zusammensetzen. Dazu benutzen Sie den Schalter \(-j\) für die Aktion "Zusammensetzen".

Nun erzeugt dpkg-split als Ausgabedatei xsnow_1:1.42-9_amd64.deb, da die Benennung den Angaben zum Paketnamen, der Paketversion sowie der Architektur (Paket-Version_Architektur.deb) folgt. Am Ende des Aufrufs geben Sie noch die Dateinamen der Stučke an, die Sie wieder zusammensetzen möchten.

Einfaches Zusammenfügen der Paketstücke

$ dpkg-split -j xsnow_1%3a1.42-9_amd64.*
Paket xsnow wird aus 4 Teilen zusammengesetzt: 1 2 3 4 fertig
$

Um eine Ausgabedatei mit einem spezifischen Dateinamen zu erhalten, kommt nun noch der Schalter \(-o\) für die Angabe der Ausgabedatei zum Einsatz. Bitte beachten Sie dabei, dass Sie im Aufruf zuerst den Schalter \(-o\) und erst danach den Schalter \(-j\) angeben. Nachfolgend sehen den vollständigen Aufruf, der als Ergebnis die Datei xsnow_1%3a1.42-9_amd64.deb liefert.

Zusammenfügen der Paketstücke mit Ausgabedatei

$ dpkg-split -o xsnow_1%3a1.42-9_amd64.deb -j xsnow_1%3a1.42-9_amd64.*
Paket xsnow wird aus 4 Teilen zusammengesetzt: 1 2 3 4 fertig
$

Anschließend raten wir Ihnen, zu überprüfen, ob alles beim Zusammenbau geklappt hat. Mittels md5sum vergleichen Sie die Hashwerte der beiden Pakete wie folgt:

Vergleichen zweier Dateien mittels md5sum
41.3.4 Keryx

Das Projekt Keryx [Keryx] beschreibt sich als eine freie, plattformunabhängige Softwareanwendung, um Linuxsysteme ohne Internetanbindung zu aktualisieren. Es hat den Fokus auf Benutzer mit Einwahlverbindungen und Systeme, die nur über eine dünne Leitung an das Internet verfügen.


Es existiert zudem eine Erweiterung um eine graphische Bedienoberfläche, die auf wxPython (Paket python-wxversion) aufsetzt.
Kapitel 42

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.

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.

42.1 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, girl1.2-gtk-3.0_3.8.5-1_i386.deb
Created, time 0.81sec, speed 75kB/sec, libgail1-3-0_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-g2a9eall1+dfsg1-2_i386.deb
Created, time 3.82sec, speed 61kB/sec, libqt4-help_4%3a4.8.5+git121-g2a9eall1+dfsg1-2_i386.deb
Created, time 8.99sec, speed 90kB/sec, libqt4-script_4%3a4.8.5+git121-g2a9eall1+dfsg1-2_i386.deb
Created, time 3.09sec, speed 82kB/sec, libqt4-scripttools_4%3a4.8.5+git121-g2a9eall1+dfsg1-2_i386.deb
Created, time 1.41sec, speed 94kB/sec, libqt4-sql_4%3a4.8.5+git121-g2a9eall1+dfsg1-2_i386.deb
Created, time 0.58sec, speed 101kB/sec, libqt4-sql-sqlite_4%3a4.8.5+git121-g2a9eall1+dfsg1-2_i386.deb
Created, time 1.52sec, speed 112kB/sec, libqt4-svg_4%3a4.8.5+git121-g2a9eall1+dfsg1-2_i386.deb
Created, time 1.05sec, speed 90kB/sec, libqt4-test_4%3a4.8.5+git121-g2a9eall1+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-xmnpatterns_4%3a4.8.5+git121-g2a9eall1+dfsg1-2_i386.deb
Created, time 3.49sec, speed 61kB/sec, libqtdbus4_4%3a4.8.5+git121-g2a9eall1+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-g2a9eall1+dfsg1-1_4%3a4.8.5+git121-g2a9eall1+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-g2a9eall1+dfsg1-2_i386.debdelta
Downloaded, time 0.51sec, speed 3708B/sec, libavresample_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-g2a9eall1+dfsg1-2_i386.debdelta
Downloaded, time 0.46sec, speed 5070B/sec, libavdevice53_6%3a9.8-2+b2_6%3a9.10-1_i386.debdelta
Created, time 0.41sec, speed 5990B/sec, libavutil115_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-g2a9eall1+dfsg1-1_4%3a4.8.5+git121-g2a9eall1+dfsg1-2_i386.debdelta
Downloaded, time 0.62sec, speed 9kB/sec, libxml-compile-cache-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-library-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-g2a9eal1+dfsfg-1_4%3 ← debdelta
Downloaded, time 5.24sec, speed 3954B/sec, libqt4-declarative_4%3a4.8.5+git121-g2a9eal1+dfsfg-1_4%3 ← debdelta
Created, time 31.47sec, speed 120kB/sec, qt4-dev-tools_4%3a4.8.5+git121-g2a9eal1+dfsfg-1-2 ← _i386.deb
Created, time 11.64sec, speed 72kB/sec, qt4-linguist-tools_4%3a4.8.5+git121-g2a9eal1+dfsfg-1-2 ← _i386.deb
Created, time 1.50sec, speed 86kB/sec, qt4-qtconfig_4%3a4.8.5+git121-g2a9eal1+dfsfg-1-2 ← _i386.deb
Downloaded, time 21.87sec, speed 1090B/sec, libgltk-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-g2a9eal1+dfsfg-1_4%3 ← debdelta
Downloaded, time 4.94sec, speed 6927B/sec, libqt4-qt3support_4%3a4.8.5+git121-g2a9eal1+dfsfg-1_4%3 ← debdelta
Created, time 15.00sec, speed 39kB/sec, qtcore4-l10n_4%3a4.8.5+git121-g2a9eal1+dfsfg-1_4%3 ← 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-g2a9eal1+dfsfg-1_4%3 ← all.deb
Downloaded, time 2.64sec, speed 16kB/sec, librose-db-object-perl_1%3a0.806-1_i386.deb
Created, time 4.11sec, speed 80kB/sec, libqt4-opengl_4%3a4.8.5+git121-g2a9eal1+dfsfg-1-2 ← _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-g2a9eal1+dfsfg-1_4%3 ← debdelta
Created, time 1.74sec, speed 40kB/sec, librty-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, libavutil152_6%3a9.10-1_i386.deb
Downloaded, time 5.43sec, speed 14kB/sec, libqt4-dev_4%3a4.8.5+git121-g2a9eal1+dfsfg-1_4%3 ← a4.8.5+git121-g2a9eal1+dfsfg-1_4%3 ← all.deb
Created, time 8.74sec, speed 65kB/sec, libqt4-network_4%3a4.8.5+git121-g2a9eal1+dfsfg-1-2 ← _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, libwscalwe2_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-g2a9eal1+dfsfg-1-2 ← _i386.deb
Downloaded, time 14.62sec, speed 15kB/sec, libqt4-designer_4%3a4.8.5+git121-g2a9eal1+dfsfg-1_4%3 ← _i386.deb
Created, time 20.09sec, speed 54kB/sec, libqt4-declarative_4%3a4.8.5+git121-g2a9eal1+dfsfg-1_4%3 ← _i386.deb
Error: applying of delta for libgltk-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-g2a9eal1+dfsfg-1-2 ← _i386.deb
Created, time 14.29sec, speed 73kB/sec, libqt4-qt3support_4%3a4.8.5+git121-g2a9eal1+dfsfg-1_4%3 ← _i386.deb
Downloaded, time 36.54sec, speed 15kB/sec, libav-tools_6%3a9.8-2+b2_6%3a9.10-1_i386. ← debdelta
Debian-Paketmanagement

42.2 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:

Konfigurationseinstieg 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 43

Paketverwaltung hinter einer Firewall

• Ausgangspunkt: abgesichertes Netz mit Firewall
• Ziel: Aktualisierung der Pakete auf einem System hinter der Firewall
• Fragen dazu:
  – was muss ich da beachten?
  – welche Ports benutzt APT (http/80 und https/443) ?
  – was ist bei ftp zu beachten?
Kapitel 44

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 45

Notizen


• 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 46

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 47

Ein eigenes Debian-Repository aufbauen

• siehe dazu: Michael Stapelberg: Kurz-Howto: Eigenes Debian-Repository aufbauen [Stapelberg-Debian-Repo]
Kapitel 48

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 49

Fazit / Zusammenfassung

49.1 Was können Sie jetzt?

In diesem Buch vermittelten wir ihnen ein umfangreiches Wissen. Das umfaßt insbesondere:

- das Verständnis, wie Softwarepakete unter Debian GNU/Linux organisiert sind und welche Infrastruktur bereitsteht, um diese zu benutzen
- die Fähigkeit, Softwarepakete zu finden, auszuwählen, zu installieren, zu konfigurieren, zu aktualisieren und wieder zu entfernen
- das Ermitteln der Abhängigkeiten zwischen Softwarepaketen und das manuelle Beheben fehlender oder auch fehlerhafter Abhängigkeiten zwischen diesen, so dass ein stabiler Zustand bzgl. der Installation des Betriebssystems erreicht wird
- das Aktuellhalten einer Debian GNU/Linux-Installation
- den Umgang mit Problemsituationen, die im Alltag bei der Paketverwaltung auftreten

Mit diesem Wissen betreiben Sie eine Linux-Installation panikfrei, d.h. ohne Schnappatmung, und legen damit den Grundstein für eine saubere und stabile Arbeitsumgebung.

49.2 Empfehlungen für Einsteiger

49.2.1 Mit welchem Programm zur Paketverwaltung gelingt der Einstieg am leichtesten?

Eine richtige Antwort darauf fällt schwer, weil das a) von ihrem Wissensstand abhängt, b) darauf basiert, welche Art von Benutzerschnittstelle Sie bevorzugen und c) in welcher Situation Sie sich gerade befinden, sprich: ob Sie lokal arbeiten oder auf einem entfernten Rechner eingeloggt sind. Es macht dabei einen großen Unterschied, ob Sie einen Desktop mit graphischer Oberfläche vor sich haben oder bspw. über ssh mit einem Server verbunden sind. Von daher geben wir Ihnen drei Empfehlungen — eine für die Komandozeile, eine für den interaktiven Textmodus und eine für die graphische Benutzerschnittstelle.

Komandozeile

Interaktiver Textmodus

aptitude (siehe Abschnitt 6.3.2). Nicht nur, weil es auch das einzige Programm mit dieser Benutzerschnittstelle ist, sondern auch, weil es Ihnen sehr viele Informationen und Möglichkeiten zu den einzelnen Paketen und deren Status bietet. Diese Darstellung in der Gesamtheit auf der Kommandozeile zu erreichen, gelingt nur mit einem Dutzend parallel geöffneter Terminalfenster.

Graphische Benutzerschnittstelle und webbasierte Lösungen

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“. Von webbasierten Lösungen wie Appnr (siehe Abschnitt 6.5.2) sind wir bislang nicht überzeugt und raten Ihnen daher nicht zu diesen.

49.2.2 Installations-CD/DVD oder Netzwerkleistung?

Die Ausgangsposition ist hier, ob der Rechner, auf dem Debian GNU/Linux installiert und zukünftig gepflegt werden soll, überwiegend über eine verlässliche Internetverbindung verfügt oder nicht. Daraus leitet sich ab:

• falls ja, dann ist eine Netzwerkleistung die richtige Auswahl (Erfahrungswert). Das erleichtert insbesondere die spätere Aktualisierung. Physische Medien oder Installationsabbilder sind hierbei nicht vorzuhalten. Bitte beachten Sie, daß bei einer Unterbrechung der Internetanbindung des Rechners eine Aktualisierung oder Änderung des Paketbestands nur mit größerem Aufwand möglich ist.

• falls nein, bleibt nur die Benutzung einer oder mehrerer Installations-CDs/DVDs oder Installationsabbilder. Diese müssen zur späteren Installation oder Entfernung von Paketen bereitliegen, um den Bestand der installierten Pakete konsistent zu halten. Der Rechner muss zudem über eine Möglichkeit verfügen, die Medien oder Installationsabbilder auch einbinden und benutzen zu können. Ihr Rechner muss zudem über eine Möglichkeit verfügen, die Medien oder Installationsabbilder auch einbinden und benutzen zu können. Wäre ein CD/DVD-Laufwerk, ein USB-Port oder ein Einschub für eine Speicherkarte, auf dem sich ein Installationsabbild befindet.

49.3 Empfehlungen für Fortgeschrittene und Profis

Sie verfügen über einen größeren Erfahrungsschatz und sind im Umgang mit UNIX-basierten Betriebssystemen versiert. Für den Ablauf schlagen wir Ihnen daher folgendes vor:


• klären Sie, welche Aktionen zur Paketverwaltung überhaupt anstehen, sprich: was konkret zu tun ist. Danach wählen Sie das passende Kommando aus.
Teil V
Anhang
Anhang A

Debian-Architekturen

A.1 Offizielle Architekturen

Debian 12 *Bookworm* unterstützt die folgenden Architekturen und Plattformen:

i386 (i486/i586/i686)
x86 (PC) 32-Bit. Trotz des Namens ab Debian 8 *Jessie* nur Pentium-kompatible und ab Debian 9 *Stretch* nur noch Pentium-II-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 Befehlsatz 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].

mipsel (mipsel-linux-gnu)
MIPS (Little Endian).

mips64el (mips64el-linux-gnuabi64)
64-Bit MIPS (Little-Endian) mit der N64 ABI, Hardware-Fließkommaarithmetik-Unterstützung und der MIPS64R2 Befehlsatzarchitektur.

ppc64le (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.

Gegenüber Debian 11 *Bullseye* ist diese Liste unverändert.

A.2 Architekturen, die nicht den Linux-Kernel verwenden

Keine dieser Architekturen ist momentan für stabile Veröffentlichungen von Debian relevant ("release architecture").
Debian-Paketmanagement

A.3 Veraltete Architekturen

Folgende Architekturen werden von Debian 12 *Bookworm* (und ggf. auch schon älteren stabilen Debian-Veröffentlichungen) nicht mehr unterstützt:

- **alpha (alpha-linux-gnu)**
  Alpha. Unterstützt von Debian 2.1 *Slink* bis Debian 6 *Squeeze*.

- **hppa (hppa-linux-gnu)**
  HP PA-RISC. Unterstützt von Debian 3 *Woody* bis Debian 6 *Squeeze*.

- **ia64 (ia64-linux-gnu)**
  Intel Itanium IA-64. Wurde zeitweise auch in *unstable* nicht mehr unterstützt, dann aber doch wiederbelebt.

- **m68k (m68k-linux-gnu)**
  Motorola 68k. Die Unterstützung war komplett eingestellt, wurde aber wiederbelebt.

- **powerpc (powerpc-linux-gnu)**

- **mips (mips-linux-gnu)**
  MIPS (Big Endian). War zuletzt in Debian 10 *Buster* Bestandteil einer stabilen Veröffentlichung.

Diese Architekturen sind gar nicht mehr verfügbar:

- **arm (arm-linux-gnu)**
  ARM (OABI). Seit Debian 6 *Squeeze* nicht mehr unterstützt und durch *armel* ersetzt.

- **m32**

- **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 ist Nachfolger *sparc64*. 
A.4 Architekturen, deren Unterstützung vorgesehen ist

Folgende Architekturen werden vom Debian-Ports-Projekt bereits unterstützt und auf eine Aufnahme in Debian vorbereitet:

**powerpcspe (powerpe-linux-gnuspe)**

**ppc64 (powerpc64-linux-gnu)**
64-Bit-PowerPC-Prozessoren mit VMX-Einheit.

**riscv64 (riscv64-linux-gnu)**
RISC-V ist eine offene Befehlszsatarchitektur, die auf dem bewährten Design von RISC (Reduced Instruction Set Computer) aufbaut und unter einer BSD-Lizenz für jedermann verfügbar ist.

**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-CPU's mit AMD64-Erweiterung und Intel CPU's 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.
Anhang B

Kommandos zur Paketverwaltung im Vergleich

B.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, beispielsweise mit Fedora 23, CentOS 6 und 7 sowie Ubuntu 15.10 *Wily Werewolf* sowie Debian 8 *Jessie* bis einschließlich Debian 12 *Bookworm*. Die Verfügbarkeit der genannten Werkzeuge und deren Möglichkeiten variiert recht stark [RPM-Salve] [RPM-Gite] [RPM-Canepa] [YUM-Salve] [dpkg-Kumar] [apt-Salve] [DNF-Dokumentation].

B.2 Die lokale Paketliste aktualisieren


Tabelle B.1: Lokale Paketliste aktualisieren

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT</td>
<td><code>apt update</code></td>
<td></td>
</tr>
<tr>
<td><code>apt-get</code></td>
<td><code>apt-get update</code></td>
<td></td>
</tr>
<tr>
<td>Aptitude</td>
<td><code>aptitude update</code></td>
<td></td>
</tr>
</tbody>
</table>

B.3 Neue Softwarepakete installieren

### Tabelle B.2: Installation eines Pakets

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT</td>
<td><code>apt install Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td></td>
<td><code>apt --no-download install Paketname</code></td>
<td>Paket wird dem Paketcache entnommen und nicht vom Paketmirror bezogen</td>
</tr>
<tr>
<td>apt-get</td>
<td><code>apt-get install Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td></td>
<td><code>apt-get --no-download install Paketname</code></td>
<td>Paket wird dem Paketcache entnommen und nicht vom Paketmirror bezogen</td>
</tr>
<tr>
<td>aptitude</td>
<td><code>aptitude install Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td>DNF</td>
<td><code>dnf install Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td></td>
<td><code>dnf group install Paketgruppe</code></td>
<td>alle Pakete aus der Paketgruppe installieren</td>
</tr>
<tr>
<td>dpkg</td>
<td><code>dpkg -i Paketdatei</code></td>
<td>Paket liegt lokal als Datei vor</td>
</tr>
<tr>
<td></td>
<td><code>dpkg --install Paketdatei</code></td>
<td>Paket liegt lokal als Datei vor</td>
</tr>
<tr>
<td>RPM</td>
<td><code>rpm --install Paketdatei</code></td>
<td>Paket liegt lokal als Datei vor</td>
</tr>
<tr>
<td></td>
<td><code>rpm --install --nodeps Paketdatei</code></td>
<td>Paket liegt lokal als Datei vor, Installation ohne Berücksichtigung der Paketabhängigkeiten</td>
</tr>
<tr>
<td>YUM</td>
<td><code>yum groupinstall Paketgruppe</code></td>
<td>Alle Pakete aus der Paketgruppe werden dem Paketrepository entnommen und installiert</td>
</tr>
<tr>
<td></td>
<td><code>yum install Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td></td>
<td><code>yum localinstall Paketname</code></td>
<td>Paket liegt lokal als Datei vor oder wird von einem HTTP- oder FTP-Server bezogen</td>
</tr>
<tr>
<td></td>
<td><code>yum reinstall Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen und erneut installiert</td>
</tr>
</tbody>
</table>

### B.4 Bereits bestehende Softwarepakete erneuern

Mit den nachfolgenden Aufrufen aktualisieren Sie ein Softwarepaket über die Kommandozeile. Unter „Pakete aktualisieren“ in Abschnitt 8.41 gehen wir detaillierter auf die Vorgehensweise und die begrifflichen Unterschiede bei `dpkg`, APT, `apt` und `aptitude` ein.

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt</td>
<td><code>apt install Paketname</code></td>
<td>aktualisierte Version des Pakets einspielen, sofern das Paket bereits in einer früheren Version installiert ist</td>
</tr>
<tr>
<td></td>
<td><code>apt safe-upgrade Paketname</code></td>
<td>aktualisierte Version des Pakets einspielen</td>
</tr>
<tr>
<td>apt-get</td>
<td><code>apt-get install Paketname</code></td>
<td>aktualisierte Version des Pakets einspielen, sofern das Paket bereits in einer früheren Version installiert ist</td>
</tr>
<tr>
<td></td>
<td><code>apt-get upgrade Paketname</code></td>
<td>aktualisierte Version des Pakets einspielen</td>
</tr>
<tr>
<td>aptitude</td>
<td><code>aptitude safe-upgrade Paketname</code></td>
<td>aktualisierte Version des Pakets einspielen</td>
</tr>
<tr>
<td>DNF</td>
<td><code>dnf update</code></td>
<td>alle installierten Pakete aktualisieren (veraltet)</td>
</tr>
</tbody>
</table>
Tabelle B.3: (continued)

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNF</td>
<td>dnf update Paketname</td>
<td>nur Paketname aktualisieren (veraltet)</td>
</tr>
<tr>
<td></td>
<td>dnf upgrade</td>
<td>alle installierten Pakete aktualisieren</td>
</tr>
<tr>
<td></td>
<td>dnf upgrade Paketname</td>
<td>nur Paketname aktualisieren</td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -F Paketdatei</td>
<td>aktualisiere ein bereits installiertes Paket; das Paket liegt lokal als Datei vor</td>
</tr>
<tr>
<td></td>
<td>rpm --freshen Paketdatei</td>
<td>aktualisiere ein bereits installiertes Paket; das Paket liegt lokal als Datei vor</td>
</tr>
<tr>
<td></td>
<td>rpm -U Paketdatei</td>
<td>aktualisiere ein bereits installiertes Paket oder installiere es, falls es noch nicht vorhanden ist; das Paket liegt lokal als Datei vor</td>
</tr>
<tr>
<td></td>
<td>rpm --upgrade Paketdatei</td>
<td>aktualisiere ein bereits installiertes Paket oder installiere es, falls es noch nicht vorhanden ist; das Paket liegt lokal als Datei vor</td>
</tr>
<tr>
<td>YUM</td>
<td>yum update</td>
<td>die gesamte Veröffentlichung aktualisieren</td>
</tr>
<tr>
<td></td>
<td>yum update Paketname</td>
<td>nur Paketname aktualisieren</td>
</tr>
<tr>
<td></td>
<td>yum update --security</td>
<td>nur die Sicherheitsaktualisierungen einspielen</td>
</tr>
<tr>
<td></td>
<td>yum update-to Paketname-Paketversion</td>
<td>eines oder alle Pakete bis zu einer bestimmten Version aktualisieren</td>
</tr>
<tr>
<td></td>
<td>yum upgrade</td>
<td>die gesamte Veröffentlichung aktualisieren und dabei auch die Pakete berücksichtigen, die veraltet sind</td>
</tr>
<tr>
<td></td>
<td>yum upgrade Paketname</td>
<td>nur Paketname aktualisieren und dabei auch die Pakete berücksichtigen, die veraltet sind</td>
</tr>
</tbody>
</table>

B.5 Eine frühere Version eines Softwarepakets einspielen (Downgrade)


Tabelle B.4: Downgraden eines Pakets

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNF</td>
<td>dnf dg Paketname</td>
<td>Paketname auf die frühere Version zurücksetzen</td>
</tr>
<tr>
<td></td>
<td>dnf downgrade Paketname</td>
<td>Paketname auf die frühere Version zurücksetzen</td>
</tr>
<tr>
<td>YUM</td>
<td>yum downgrade Paketname</td>
<td>Paketname auf die frühere Version zurücksetzen</td>
</tr>
</tbody>
</table>

B.6 Ein Softwarepaket erneut installieren

Mit den nachfolgenden Aufrufen installieren Sie ein Softwarepaket über die Komandozeile erneut.
### Tabelle B.5: Erneute Installation eines Pakets

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT</td>
<td><code>apt reinstall Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td>apt-get</td>
<td><code>apt-get install --reinstall Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td>aptitude</td>
<td><code>aptitude reinstall Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td>RPM</td>
<td><code>rpm --reinstall Paketdatei</code></td>
<td>installiere das Paket erneut, dieses liegt als</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lokale Datei vor</td>
</tr>
<tr>
<td>YUM</td>
<td><code>yum reinstall Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
</tbody>
</table>

### B.7 Ein Softwarepaket nur herunterladen

Mit den nachfolgenden Aufrufen laden Sie ein Softwarepaket in das aktuelle Verzeichnis herunter.

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT</td>
<td><code>apt download Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td>aptitude</td>
<td><code>aptitude download Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td>YUM</td>
<td><code>yum install --downloadonly Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen und im</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Paketcache gespeichert, üblicherweise im</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Verzeichnis /var/cache/yum/architektur/prod/repo/</td>
</tr>
<tr>
<td></td>
<td><code>yumdownloader Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen</td>
</tr>
<tr>
<td></td>
<td><code>yumdownloader --resolve Paketname</code></td>
<td>Paket samt dessen Paketabhängigkeiten wird</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dem Paketrepository entnommen und im aktuelle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Verzeichnis gespeichert</td>
</tr>
<tr>
<td></td>
<td><code>yumdownloader --destdir Verzeichnis Paketname</code></td>
<td>Paket wird dem Paketrepository entnommen und im</td>
</tr>
<tr>
<td></td>
<td></td>
<td>angegebenen Verzeichnis gespeichert</td>
</tr>
</tbody>
</table>

### B.8 Paket löschen / entfernen

Mit den nachfolgenden Aufrufen entfernen Sie ein Softwarepaket über die Kommandozeile. Unter „Pakete deinstallieren“ in Abschnitt 8.43 gehen wir detaillierter auf dpkg, APT und aptitude ein.

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt</td>
<td><code>apt remove Paketname</code></td>
<td>entfernt das Paket, die Konfigurationsdateien</td>
</tr>
<tr>
<td></td>
<td></td>
<td>des Pakets bleiben erhalten</td>
</tr>
<tr>
<td>apt-get</td>
<td><code>apt-get remove Paketname</code></td>
<td>entfernt das Paket, die Konfigurationsdateien</td>
</tr>
<tr>
<td></td>
<td></td>
<td>des Pakets bleiben erhalten</td>
</tr>
<tr>
<td></td>
<td><code>apt-get purge Paketname</code></td>
<td>entfernt das Paket inklusive der</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Konfigurationsdateien des Pakets</td>
</tr>
</tbody>
</table>
B.9 Alle installierten Pakete auflisten


Tabelle B.8: Softwarepakete auflisten

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt</td>
<td>apt list --installed</td>
<td>alle installierten Pakete auflisten</td>
</tr>
<tr>
<td>apt-cache</td>
<td>apt-cache search Paketname</td>
<td>alle verfügbaren Pakete für Paketname.auflisten, durchsucht dabei Paketname und Beschreibung</td>
</tr>
<tr>
<td>apt-cache</td>
<td>apt-cache search--names-only Paketname</td>
<td>alle verfügbaren Pakete für Paketname.auflisten, berücksichtigt dabei nur den Paketname</td>
</tr>
<tr>
<td>aptitude</td>
<td>aptitude search ‘-i’</td>
<td>alle installierten Pakete auflisten</td>
</tr>
<tr>
<td></td>
<td>aptitude versions Paketname</td>
<td>alle verfügbaren Pakete für Paketname.auflisten, auch die (noch) nicht installierten Varianten</td>
</tr>
<tr>
<td>DNF</td>
<td>dnf list installed</td>
<td>alle installierten Pakete auflisten</td>
</tr>
<tr>
<td>dpkg</td>
<td>dpkg -l</td>
<td>alle installierten Pakete anzeigen</td>
</tr>
<tr>
<td></td>
<td>dpkg --list</td>
<td>alle installierten Pakete auflisten</td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -qa</td>
<td>alle installierten Pakete auflisten</td>
</tr>
</tbody>
</table>
Tabelle B.8: (continued)

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>rpm</td>
<td>rpm -qa --last</td>
<td>alle installierten Pakete auflisten, dabei die zuletzt installierte Pakete zuerst</td>
</tr>
<tr>
<td></td>
<td>rpm --query --all</td>
<td>alle installierten Pakete auflisten</td>
</tr>
<tr>
<td></td>
<td>rpm --query --all --last</td>
<td>alle installierten Pakete auflisten, dabei die zuletzt installierte Pakete zuerst</td>
</tr>
<tr>
<td>YUM</td>
<td>yum list Paketname</td>
<td>anzeigen, welche Versionen des Pakets installiert sind</td>
</tr>
<tr>
<td></td>
<td>yum list all</td>
<td>alle installierten und verfügbaren Pakete auflisten</td>
</tr>
<tr>
<td></td>
<td>yum list available</td>
<td>alle verfügbaren Pakete auflisten</td>
</tr>
<tr>
<td></td>
<td>yum list installed</td>
<td>alle installierten Pakete auflisten</td>
</tr>
<tr>
<td></td>
<td>yum list updates</td>
<td>alle aktualisierbaren Pakete auflisten</td>
</tr>
</tbody>
</table>

B.10 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 B.9: Einzelnes Softwarepaket auflisten

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt</td>
<td>apt list Paketname</td>
<td>Informationen und den Installationsstatus zu Paketname anzeigen</td>
</tr>
<tr>
<td></td>
<td>apt show Paketname</td>
<td>Detaillierte Informationen und den Installationsstatus zu Paketname anzeigen</td>
</tr>
<tr>
<td>aptitude</td>
<td>aptitude show Paketname</td>
<td></td>
</tr>
<tr>
<td>DNF</td>
<td>dnf info Paketname</td>
<td>Informationen zu Paketname anzeigen</td>
</tr>
<tr>
<td></td>
<td>dnf list installed</td>
<td>alle installierten Pakete anzeigen</td>
</tr>
<tr>
<td></td>
<td>dnf list installed Paketname</td>
<td>Installationsstatus zu Paketname anzeigen</td>
</tr>
<tr>
<td>dpkg</td>
<td>dpkg -l Paketname</td>
<td>Ausgabe des Installationsstatus</td>
</tr>
<tr>
<td></td>
<td>dpkg --list Paketname</td>
<td>Ausgabe des Installationsstatus</td>
</tr>
<tr>
<td></td>
<td>dpkg --s Paketname</td>
<td>Ausgabe der Paketinformationen</td>
</tr>
<tr>
<td></td>
<td>dpkg --status Paketname</td>
<td>Ausgabe der Paketinformationen</td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -q Paketname</td>
<td>Ausgabe des Installationsstatus für Paketname</td>
</tr>
<tr>
<td></td>
<td>rpm -qi Paketname</td>
<td>Informationen zu einem installierten Paket anzeigen</td>
</tr>
<tr>
<td></td>
<td>rpm --query Paketname</td>
<td>analog zu -q</td>
</tr>
<tr>
<td></td>
<td>rpm --query --info Paketname</td>
<td>analog zu -qi</td>
</tr>
<tr>
<td></td>
<td>rpm --query --package Paketdatei</td>
<td>analog zu -qp</td>
</tr>
<tr>
<td>YUM</td>
<td>yum info Paketname</td>
<td>Informationen zu dem Paket anzeigen</td>
</tr>
</tbody>
</table>

B.11 Abhängigkeiten anzeigen

Tabelle B.10: Paketabhängigkeiten anzeigen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpkg</td>
<td>dpkg -f <code>Paketdatei</code> Depends</td>
<td>das Paket muß dazu lokal als Datei vorliegen</td>
</tr>
<tr>
<td>dpkg-deb</td>
<td>dpkg-deb -f <code>Paketdatei</code> Depends</td>
<td>das Paket muß dazu lokal als Datei vorliegen</td>
</tr>
<tr>
<td>APT</td>
<td>apt-cache depends <code>Paketname</code></td>
<td>umgekehrte Abhängigkeiten anzeigen</td>
</tr>
<tr>
<td></td>
<td>apt-cache rdepends <code>Paketname</code></td>
<td>Abhängigkeiten anzeigen</td>
</tr>
<tr>
<td>apt-rdepends</td>
<td>apt-rdepends -r <code>Paketname</code></td>
<td>Abhängigkeiten anzeigen</td>
</tr>
<tr>
<td>aptitude</td>
<td>aptitude search <code>~R</code> <code>Paketname</code></td>
<td>Abhängigkeiten anzeigen</td>
</tr>
<tr>
<td>grep-status</td>
<td>grep-status -F Package -s  <code>Depends Paketname</code></td>
<td>Abhängigkeiten anzeigen</td>
</tr>
<tr>
<td></td>
<td>grep-status -F <code>Package</code> -s <code>Depends Paketname</code></td>
<td>Abhängigkeiten anzeigen</td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -qR <code>Paketname</code></td>
<td>das Paket muß lokal auf dem System installiert sein</td>
</tr>
<tr>
<td></td>
<td>rpm --query --requires <code>Paketname</code></td>
<td>das Paket muß lokal auf dem System installiert sein</td>
</tr>
<tr>
<td></td>
<td>rpm -qR <code>Paketdatei</code></td>
<td>das Paket muß dazu lokal als Datei vorliegen</td>
</tr>
<tr>
<td></td>
<td>rpm --query --package --requires <code>Paketdatei</code></td>
<td>das Paket muß dazu lokal als Datei vorliegen</td>
</tr>
<tr>
<td>YUM</td>
<td>yum deplist <code>Paketname</code></td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
<tr>
<td></td>
<td>yum info <code>Paketname</code></td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
<tr>
<td></td>
<td>repoquery --requires <code>Paketname</code></td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
<tr>
<td></td>
<td>yumdownloader --resolve <code>Paketname</code></td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
</tbody>
</table>

B.12 Alle Dateien eines Pakets anzeigen, bevor es installiert wird


Tabelle B.11: Paketinhalte vor der Installation anzeigen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt-file</td>
<td>apt-file list <code>Paketname</code></td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
<tr>
<td>DNF</td>
<td>dnf repoquery -l <code>Paketname</code></td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
<tr>
<td>dpkg</td>
<td>dpkg --contents <code>Paketname</code></td>
<td>Dateiliste der lokalen Paketdatei anzeigen</td>
</tr>
<tr>
<td>dpkg-deb</td>
<td>dpkg-deb -c <code>Paketname</code></td>
<td>Dateiliste der lokalen oder entfernten Paketdatei anzeigen</td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -ql <code>Paketname</code></td>
<td>Dateiliste der lokalen oder entfernten Paketdatei anzeigen</td>
</tr>
<tr>
<td></td>
<td>rpm --query --list <code>Paketname</code></td>
<td>Dateiliste der lokalen oder entfernten Paketdatei anzeigen</td>
</tr>
<tr>
<td></td>
<td>rpm --query --package --list <code>Paketdatei</code></td>
<td>Dateiliste der lokalen oder entfernten Paketdatei anzeigen</td>
</tr>
<tr>
<td>YUM</td>
<td>repoquery -l <code>Paketname</code></td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
<tr>
<td></td>
<td>repoquery --list <code>Paketname</code></td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
</tbody>
</table>
B.13 Alle Dateien eines installierten Pakets anzeigen


Tabelle B.12: Paketinhalte anzeigen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt-file</td>
<td>apt-file list Paketname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>apt-file show Paketname</td>
<td></td>
</tr>
<tr>
<td>DNF</td>
<td>dnf repoquery -l Paketname</td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
<tr>
<td>dpkg</td>
<td>dpkg -L Paketname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dpkg --listfiles Paketname</td>
<td></td>
</tr>
<tr>
<td>dpkg-query</td>
<td>dpkg-query -L Paketname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dpkg-query --listfiles Paketname</td>
<td></td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -ql Paketname</td>
<td>Dateiliste der lokalen Paketdatei anzeigen</td>
</tr>
<tr>
<td></td>
<td>rpm --query --list Paketname</td>
<td>Dateiliste der lokalen Paketdatei anzeigen</td>
</tr>
<tr>
<td>YUM</td>
<td>repoquery -l Paketname</td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
<tr>
<td></td>
<td>repoquery --list Paketname</td>
<td>Dateiliste zu dem Paket anzeigen</td>
</tr>
</tbody>
</table>

B.14 Alle Konfigurationsdateien eines Pakets anzeigen

dpkg, dpkg-query und rpm listen nur die Einträge auf, die sich bereits im Paket befinden und nicht diese, die erst zur Laufzeit angelegt werden. Für dpkg gehen wir darauf in „Konfigurationsdateien eines Pakets anzeigen“ (siehe Abschnitt 8.30) genauer ein.

Tabelle B.13: Konfigurationsdateien eines Pakets anzeigen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpkg</td>
<td>dpkg -s Paketname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dpkg --status Paketname</td>
<td></td>
</tr>
<tr>
<td>dpkg-query</td>
<td>dpkg-query -s Paketname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dpkg-query --status Paketname</td>
<td></td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -qc Paketname</td>
<td>listet alle Dateien auf, die im rpm-Paket als Konfiguration gekennzeichnet sind</td>
</tr>
<tr>
<td></td>
<td>rpm --query --configfiles Paketname</td>
<td>listet alle Dateien auf, die im rpm-Paket als Konfiguration gekennzeichnet sind</td>
</tr>
</tbody>
</table>

Alternativ listen Sie nur die Dateien auf, die sich im Verzeichnis /etc befinden, hier anhand des Beispelpakets paket.rpm:

```
rpm -qc paket.rpm | grep "/etc"
```

B.15 Alle Dokumentationsdateien eines Pakets anzeigen

dpkg und apt können das nicht. Daher geben wir nur den Aufruf von rpm an.
### Tabelle B.14: Dokumentationsdateien eines Pakets anzeigen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM</td>
<td>rpm -qd <strong>Paketname</strong></td>
<td>listet alle Dateien auf, die im rpm-Paket explizit als Dokumentation gekennzeichnet sind</td>
</tr>
<tr>
<td></td>
<td>rpm --query --docfiles <strong>Paketname</strong></td>
<td>listet alle Dateien auf, die im rpm-Paket explizit als Dokumentation gekennzeichnet sind</td>
</tr>
</tbody>
</table>

Alternativ listen Sie nur die Dateien auf, die sich im Verzeichnis `/usr/share/doc` befinden, hier anhand des Beispielpakets `paket.rpm`:

```
rpm -qd paket.rpm | grep "^/usr/share/doc"
```

### B.16 Paket identifizieren, aus dem eine Datei stammt


### Tabelle B.15: Paket zu Datei finden

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt-file</td>
<td>apt-file find <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in allen verfügbaren Paketen</td>
</tr>
<tr>
<td></td>
<td>apt-file search <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in allen verfügbaren Paketen</td>
</tr>
<tr>
<td>DNF</td>
<td>dnf provides <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in allen verfügbaren Paketen. <strong>Dateiname</strong> umfaßt hier den vollständigen Namen inklusive Pfad</td>
</tr>
<tr>
<td>dpkg</td>
<td>dpkg -S <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in den installierten Paketen</td>
</tr>
<tr>
<td></td>
<td>dpkg --search <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in den installierten Paketen</td>
</tr>
<tr>
<td>dpkg-query</td>
<td>dpkg-query -S <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in den installierten Paketen</td>
</tr>
<tr>
<td></td>
<td>dpkg-query --search <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in den installierten Paketen</td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -qf <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in den installierten Paketen. <strong>Dateiname</strong> umfaßt hier den vollständigen Namen inklusive Pfad</td>
</tr>
<tr>
<td></td>
<td>rpm --query --file <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in den installierten Paketen. <strong>Dateiname</strong> umfaßt hier den vollständigen Namen inklusive Pfad</td>
</tr>
<tr>
<td>YUM</td>
<td>yum provides <strong>Dateiname</strong></td>
<td>Suche nach dem Dateinamen in allen verfügbaren Paketen. <strong>Dateiname</strong> umfaßt hier den vollständigen Namen inklusive Pfad</td>
</tr>
</tbody>
</table>
B.17 Paketstatus anzeigen


<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt-cache</td>
<td>apt-cache show Paketname</td>
<td>Suche in allen verfügbaren Paketen</td>
</tr>
<tr>
<td>aptitude</td>
<td>aptitude show Paketname</td>
<td>Suche in allen verfügbaren Paketen</td>
</tr>
<tr>
<td>dpkg</td>
<td>dpkg --status Paketname</td>
<td>Paketname muß lokal installiert sein</td>
</tr>
<tr>
<td></td>
<td>dpkg -I Dateiname</td>
<td>Dateiname bezeichnet eine lokale Datei</td>
</tr>
<tr>
<td>dpkg-query</td>
<td>dpkg-query --status Paketname</td>
<td>Paketname muß lokal installiert sein</td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -qi Paketname</td>
<td>Paketname muß lokal installiert sein</td>
</tr>
<tr>
<td></td>
<td>rpm --query --info Paketname</td>
<td>Paketname muß lokal installiert sein</td>
</tr>
<tr>
<td></td>
<td>rpm --query --info --package Dateiname</td>
<td>Dateiname muß lokal vorliegen</td>
</tr>
<tr>
<td>YUM</td>
<td>yum info Paketname</td>
<td>Paketname muß lokal installiert sein</td>
</tr>
</tbody>
</table>

B.18 Aktualisierbare Pakete anzeigen


<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt</td>
<td>apt list --upgradable</td>
<td>alle Pakete auflisten, für die eine neue Version bereitsteht</td>
</tr>
<tr>
<td></td>
<td>apt list --upgradable Paketname</td>
<td>anzeigen, ob für das Paket eine neue Version bereitsteht</td>
</tr>
<tr>
<td>apt-get</td>
<td>apt-get upgrade -u</td>
<td>alle Pakete auflisten, für die eine neue Version bereitsteht</td>
</tr>
<tr>
<td></td>
<td>apt-get upgrade --show-upgraded</td>
<td>analog zu -u (Langform)</td>
</tr>
<tr>
<td></td>
<td>apt-get upgrade -u -s</td>
<td>Simulation, analog zu -u</td>
</tr>
<tr>
<td></td>
<td>apt-get upgrade --show-upgraded --simulate</td>
<td>Simulation, analog zu -u -s (Langform)</td>
</tr>
<tr>
<td>aptitude</td>
<td>aptitude search ‘-U’</td>
<td>alle aktualisierbaren Pakete anzeigen</td>
</tr>
<tr>
<td>DNF</td>
<td>dnf list upgrades</td>
<td>alle aktualisierbaren Pakete anzeigen</td>
</tr>
<tr>
<td>YUM</td>
<td>yum updateinfo list available</td>
<td>Auflistung der Aktualisierungen für die bereits installierten Pakete</td>
</tr>
<tr>
<td></td>
<td>yum updateinfo list sec</td>
<td>alle Sicherheitsaktualisierungen für die bereits installierten Pakete anzeigen</td>
</tr>
</tbody>
</table>
Tabelle B.17: (continued)

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>yum</td>
<td><code>yum updateinfo list --security</code></td>
<td>alle Sicherheitsaktualisierungen für die bereits installierten Pakete anzeigen</td>
</tr>
<tr>
<td>Zypper</td>
<td><code>zypper list updates --all</code></td>
<td>alle aktualisierbaren Pakete anzeigen</td>
</tr>
</tbody>
</table>

**B.19 Verfügbare Pakete anzeigen**


Tabelle B.18: Verfügbare Pakete anzeigen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt</td>
<td><code>apt list</code></td>
<td>alle verfügbaren (bekannten) Pakete samt Status auflisten</td>
</tr>
<tr>
<td>apt-cache</td>
<td><code>apt-cache pkgnames</code></td>
<td>alle verfügbaren (bekannten) Pakete auflisten</td>
</tr>
<tr>
<td>DNF</td>
<td><code>dnf group list</code></td>
<td>alle Paketgruppen auflisten</td>
</tr>
<tr>
<td>DNF</td>
<td><code>dnf list --available</code></td>
<td>alle verfügbaren Pakete anzeigen</td>
</tr>
<tr>
<td>DNF</td>
<td><code>dnf repository-packages repo list --available</code></td>
<td>alle verfügbaren Pakete aus dem Repository repo anzeigen</td>
</tr>
<tr>
<td>YUM</td>
<td><code>yum list available</code></td>
<td>alle verfügbaren Pakete anzeigen</td>
</tr>
</tbody>
</table>

**B.20 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 8.32.1 und „Paket auf Veränderungen prüfen“ in Abschnitt 8.32.

Tabelle B.19: Paketsignatur überprüfen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>debsums</td>
<td><code>debsums Paketname</code></td>
<td>alle Dateien des angegebenen Paketes überprüfen</td>
</tr>
<tr>
<td>dpkg</td>
<td><code>dpkg -V Paketname</code></td>
<td>alle Dateien des angegebenen Paketes überprüfen</td>
</tr>
<tr>
<td>dpkg-sig</td>
<td><code>dpkg-sig --verify Paketname</code></td>
<td>GnuPG-Signatur des Pakets prüfen</td>
</tr>
<tr>
<td>gpg</td>
<td><code>gpg --verify Paketname</code></td>
<td>GnuPG-Signatur des Pakets prüfen</td>
</tr>
<tr>
<td>RPM</td>
<td><code>rpm -K Paketname</code></td>
<td>GnuPG-Signatur des Pakets prüfen</td>
</tr>
<tr>
<td></td>
<td><code>rpm --checksig Paketname</code></td>
<td>GnuPG-Signatur des Pakets prüfen</td>
</tr>
</tbody>
</table>
B.21 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 B.20: Paket auf Veränderungen prüfen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpkg</td>
<td>dpkg -V</td>
<td>prüft alle installierten Pakete</td>
</tr>
<tr>
<td></td>
<td>dpkg --verify Paketname</td>
<td>prüft nur das angegebene Paket</td>
</tr>
<tr>
<td>RPM</td>
<td>rpm -V</td>
<td>prüft alle installierten Pakete</td>
</tr>
<tr>
<td></td>
<td>rpm -Vp Paketname</td>
<td>prüft nur das angegebene Paket</td>
</tr>
<tr>
<td></td>
<td>rpm --verify --all</td>
<td>prüft alle installierten Pakete</td>
</tr>
<tr>
<td></td>
<td>rpm --verify --package Paketname</td>
<td>prüft nur das angegebene Paket</td>
</tr>
<tr>
<td>YUM</td>
<td>yum verify Paketname</td>
<td>Paketname prüfen; erfordert YUM-Plugin yum-verify</td>
</tr>
</tbody>
</table>

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 8.32 genauer ein.

B.22 Transaktionshistorie anzeigen

dpkg, apt und aptitude besitzen keine expliziten Schalter dafür. Im Abschnitt „Liste der zuletzt installierten Pakete anzeigen“ in Abschnitt 8.18 zeigen wir Ihnen, wie Sie das über die Logdateien lösen. DNF und YUM erlauben es hingegen, die Transaktionshistorie darzustellen. Dabei helfen Ihnen diese Kommandos:

Tabelle B.21: Transaktionshistorie anzeigen

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNF</td>
<td>dnf history</td>
<td>die gesamte Historie aller Transaktionen anzeigen</td>
</tr>
<tr>
<td></td>
<td>dnf history list</td>
<td>die gesamte Historie aller Transaktionen anzeigen</td>
</tr>
<tr>
<td>YUM</td>
<td>yum history</td>
<td>die gesamte Historie aller Transaktionen anzeigen</td>
</tr>
<tr>
<td></td>
<td>yum history info Paketname</td>
<td>detailliertere Transaktionen zu Paketname anzeigen</td>
</tr>
<tr>
<td></td>
<td>yum history list all</td>
<td>die gesamte Historie aller Transaktionen anzeigen</td>
</tr>
<tr>
<td></td>
<td>yum history package-list Paketname</td>
<td>die gesamte Historie der Transaktionen für das Paket Paketname anzeigen</td>
</tr>
<tr>
<td></td>
<td>yum history packages-list Paketname1 Paketname2</td>
<td>die gesamte Historie der Transaktionen für die beiden Pakete Paketname1 und Paketname2 anzeigen</td>
</tr>
<tr>
<td></td>
<td>yum history summary Paketname</td>
<td>Zusammenfassung der Transaktionen zu Paketname</td>
</tr>
</tbody>
</table>
Eine installierte Veröffentlichung aktualisieren (Distributionsupgrade) ==
Mit den nachfolgenden Kommandos aktualisieren Sie eine bestehende Installation:

Tabelle B.22: Distributionsupgrade

<table>
<thead>
<tr>
<th>Werkzeug</th>
<th>Aufruf</th>
<th>Anmerkungen</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt</td>
<td>apt full-upgrade</td>
<td>Aktualisiere alle bereits installierten Softwarepakete auf die aktuellste, verfügbare Version</td>
</tr>
<tr>
<td>apt-get</td>
<td>apt-get dist-upgrade</td>
<td>Aktualisiere alle bereits installierten Softwarepakete auf die aktuellste, verfügbare Version</td>
</tr>
<tr>
<td>aptitude</td>
<td>aptitude full-upgrade</td>
<td>Aktualisiere alle bereits installierten Softwarepakete auf die aktuellste, verfügbare Version</td>
</tr>
<tr>
<td></td>
<td>aptitude safe-upgrade</td>
<td>Aktualisiere alle bereits installierten Softwarepakete auf die aktuellste, verfügbare Version mit Ausnahme von ungenutzten Paketen</td>
</tr>
</tbody>
</table>
Anhang C

Paketformat im Einsatz

C.1 Embedded-Geräte

**Armbian**
Debian für ARM-basierte Geräte, bspw. ein Cubieboard oder Cubietruck [Armbian]

**Raspbian**
Debian GNU/Linux so rekompliiert, dass er mit bestöglicher Optimierung auch auf den schwächsten Raspberry Pis läuft. [Raspbian]

**Raspberry Pi OS**

C.2 Bildung

**Skolelinux**
früher DebianEdu, für den Einsatz in Schulen [Skolelinux]

**Edubuntu**
Ubuntu für Schule und Bildung [Edubuntu]

**LernStick**
Live-Distribution auf einem USB-Stick [LernStick]

C.3 Desktop

**Deepin**
Angeblich die meistgenutzte Linux-Distribution Chinas [Deepin]

**Emmabuntüs**

**LiMux**
Linux-Distribution, die im Rahmen der Migration der Stadtverwaltung München zum Einsatz kam (2009-2020) [LiMux]
Linux Mint, Linux Mint Debian Edition (LMDE)
basiert auf Ubuntu bzw. Debian testing [LinuxMint]

PureOS
Linux-Distribution für Smartphones und Notebooks der Firma Purism. [PureOS], [Purism]

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]

C.4 Live-CD

Aptosid
Linux-Distributionen für Desktop-Computer und Notebooks, basierend auf Debian unstable [Aptosid]

BackTrack, Kali Linux

Debian Live System
Projekt zur Erstellung und Pflege von Debian-basierten Live-Systemen und Debian Live Images [DebianLiveSystem]

Finnix
Debian-Minimalsystem, gedacht für Systemadministratoren zur Wartung und Reparatur im Rahmen von Notfällen [Finnix]

Grml
textbasiert, gedacht als Rettungssystem und für die Systemanalyse, basiert auf Debian unstable [Grml]

Knoppix
Mutter aller Live-Systeme mit dem Vater Klaus Knopper. Es basiert auf einer Mischung aus Debian stable, unstable und testing sowie zusätzlichen, nicht-freien Komponenten [Knoppix]

MX Linux
Linux-Distributionen für Desktop-Computer und Notebooks, basierend auf Debian stable [MXLinux]

Siduction
Linux-Distributionen für Desktop-Computer und Notebooks, basierend auf Debian unstable [Siduction]

Slax
Live-Distribution mit dem Fokus auf Anpassbarkeit an die eigenen Bedürfnisse [Slax].

Tails
The Amnesic Incognito Live System (TAILS) [TAILS] — eine speziell auf Privatsphäre ausgelegte Debian-Distribution. TAILS kommt beim Tor-Projekt [TorProject] zum Einsatz.

Nach aktuellem Stand lassen sich zumindest Aptosid, Siduction, Grml und Knoppix ebenfalls auf der Festplatte installieren und sind damit nicht nur Live-CDS.

C.5 Minimalsysteme

Damn Small Linux
wird seit Mitte 2012 nicht mehr weiterentwickelt [DamnSmallLinux]
Finnix
Debian-Minimalsystem, gedacht für Systemadministratoren zur Wartung und Reparatur im Rahmen von Notfällen [Finnix]

Tiny Core Linux
Nachfolger von Damn Small Linux. Die dCore-Variante basiert auf der Infrastruktur und den Paketen von Debian und Ubuntu [TinyCoreLinux]

C.6 Spieleplattform

Drauger OS
Betriebssystem für Linux Desktop Gaming auf der Basis von Ubuntu LTS [DraugerOS]

Steam OS

C.7 Mobile Architekturen


C.8 Anstatt Linux


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.

C.9 Nachbauten und Derivate


C.10  Weitere Debian-Derivate

Einen ausführlichen Überblick zu weiteren Debian-Derivaten gibt der Debian-Derivate-Zensus. Er ist ein Bestandteil des Debian-Wikis [DebianDerivativeCensus].
**Anhang D**

**Früher im Buch erwähnte Werkzeuge**

Das Universum der Werkzeuge für Debian GNU/Linux unterliegt stetigen Veränderungen und entwickelt sich weiter. Eine ganze Reihe davon haben wir in früheren Versionen dieses Buches besprochen, inzwischen wurden diese jedoch obsolet. Das betrifft die folgenden Werkzeuge:

**apprecommender**

**aptoncd**
Werkzeug, um Aktualisierungen auf Systemen ohne Netzwerkanschluss einzuspielen. `aptoncd` war zuletzt in Debian 8 `Jessie` enthalten und wurde 2018 aus Debian `unstable` entfernt.

**aptsh**

**ara** und **xara-gtk**

**auto-apt**
Programm ähnlich zu `command-not-found`, welches jedoch ein nicht installiertes Programm, dass versucht wurde zu benutzen, gleich installiert. (Passende Benutzerrechte vorausgesetzt.). `auto-apt` war zuletzt in Debian 8 `Jessie` enthalten und wurde 2017 aus Debian `unstable` entfernt.

**debtags-edit**

**gdebi-kde**
Eine graphische, KDE-basierte Benutzerschnittstelle zu Gdebi (siehe Abschnitt 6.4.5). `gdebi-kde` war zuletzt in Debian 9 `Stretch` enthalten. Es war das letzte Paket, dass noch die Bibliothek `pykde4` benötigte.

**gui-apt-key und curses-apt-key**
Das Werkzeug `apt-key` wurde abgekündigt. In Folge werden auch dessen alternative Bedienschnittstellen `gui-apt-key` [Debian-Paket-gui-apt-key] und `curses-apt-key` [curses-apt-key] nicht mehr weiterentwickelt. `gui-apt-key` waren zuletzt in Debian 9 `Stretch` enthalten.
**software-center**

Anhang E

Literaturverzeichnis und Referenzen

- [apt4rpm] apt4rpm: http://apt4rpm.sourceforge.net/
• [apt-dater-Projektseite] Projektseite zu apt-dater, https://www.ihb.de/apt-dater/
• [apt-mirror-Projektseite] Projektseite zu apt-mirror, http://apt-mirror.github.io/
• [Arch-Linux-pacapt] pacapt auf GitHub: https://github.com/icy/pacapt
• [berliOS] berliOS, https://www.berflios.de/
• [Click-Paket-Format] Canonicals Click Paketformat, https://click.readthedocs.org/en/latest/
• [CLT] Chemnitzer Linux-Tage, https://chemnitzer.linux-tage.de/
• [comptia-linux] CompTIA Linux+, https://www.comptia.org/certifications/linux
• [cobbler-webseite] Cobbler, https://cobbler.github.io/
• [CreativeCommons] Creative Commons Namensnennung — Weitergabe unter gleichen Bedingungen 4.0 International Lizenz, https://creativecommons.org/licenses/by-sa/4.0/
• [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.debian.org/675199
• [curses-apt-key-rt] Intent to package curses-apt-key, https://bugs.debian.org/675187
• [DebConf] Debian Entwicklerkonferenz (DebConf), https://www.debconf.org/
• [DebianArchiveKit] Debian Archive Kit (dak), https://salsa.debian.org/ftp-team/dak
• [Debianforum-Wiki-Backports] Debian Backports im Debianforum Wiki: https://wiki.debianforum.de/Backports
• [Debian-Backports] Debian Backports: https://backports.debian.org/
• [Debian-Debtags-Editor] Debtags Editor, https://debtags.debian.net/edit/
• [Debian-Donations] Spenden an Debian, https://www.debian.org/donations
• [Debian-Fasttrack] Debian Fasttrack, https://fasttrack.debian.net/
• [Debian-Paket-checkinstall] Debian-Paket *checkinstall*, https://packages.debian.org/de/bookworm/checkinstall
• [Debian-Paket-smartpm] Debian-Paket smartpm, https://packages.debian.org/de/buster/smartpm
• [Debian-Paketierbuch] Debian-Paketierbuch (dbp), https://salsa.debian.org/ddp-team/dpb
• [Eintrag von Fastly auf der Debian-Partner-Liste, https://www.debian.org/partners/#Fastly
• [Debian-Sources-List-Generator] Debian Sources List Generator, https://debgen.simplylinux.ch/
• [DilOS] DilOS, http://www.dilos.org/
• [Docker] Docker, https://www.docker.com/
• [dpkg-Kumar] Avishek Kumar: 15 Practical Examples of "dpkg commands" for Debian Based Distros, http://www.tecmint.com/dpkg-command-examples/
• [DysonOS] Dyson OS, https://www.osdyson.org/
• [GObject-Introspection] GObject Introspection Middleware, https://wiki.gnome.org/Projects/GObjectIntrospection
• [Java-Apt] Java Annotation Processing Tool, https://metro.java.net/1.5/docs/apt.html
• [Krafft-Debian-System144] Ebd., S. 144 ff.
• [Krafft-Debian-System137ff] Ebd., Kapitel 5, S. 137-294
• [Trainings der Linux Foundation], https://training.linuxfoundation.org/
• [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, https://www.linuxmint.com/
• [localepurge] localepurge, Projektseite im Linux Wiki, http://linuxwiki.de/localepurge
• [LXC] Linux Containers (LXC), https://linuxcontainers.org/lxc/introduction/
• [MXLinux] MX Linux, https://mxlinux.org/
• [Neo900] Neo900-Projekt, https://neo900.org/
• [PCLinuxOS] PCLinuxOS, https://www.pclinuxos.com/
• [pfsense] pfSense, https://www.pfsense.org/
• [PureOS] PureOS, https://www.pureos.net/
Debian-Paketmanagement

- [Sentinel4Mobile] Sentinel4Mobile Berlin, Werner Heuser, http://sentinel4mobile.de/
- [SingleClickInstall] https://wiki.ubuntu.com/SingleClickInstall
- [Skolelinux] Skolelinux, https://skolelinux.de/
- [SmartPM] Smart Package Manager, Projektseite, https://labix.org/smart
- Entfernung von SmartPM aus Debian, https://bugs.debian.org/946692
- [SourceForge] SourceForge, https://sourceforge.net/
• [SteamOS] Steam OS, http://store.steampowered.com/steamos/
• [SWITCH] SWITCH, das Hochleistungsnetzwerk der Schweizer Hochschulen, https://www.switch.ch/
• [sysgetGitHub] sysget auf GitHub, https://github.com/emilengler/sysget
• [TinyCoreLinux] Tiny Core Linux, http://tinycorelinux.net/
• [TorProject] Tor Project, https://www.torproject.org/
• [UbuntuBSD] UbuntuBSD, https://www.ubuntubsd.org/
• [Ubuntu-Lauchpad] Ubuntu Launchpad, https://launchpad.net/ubuntu
• [Ubuntu-One] Ubuntu One, http://ubuntuone.com
• [Ubuntu-Paket-apt-clone] Ubuntu-Paket apt-clone, https://launchpad.net/apt-clone
• [Ubuntu-Paket-ubumirror] Ubuntu-Paket ubumirror, https://launchpad.net/ubumirror
• [Ubuntu-Snappy-Projekt] Ubuntu Package Format Snappy (Projektseite), http://snapcraft.io/
• [Ubuntu-Software-Center] Ubuntu Software Center, Projektseite/Wiki, https://wiki.ubuntu.com/SoftwareCenter
• [Ubuntu-Sources-List-Generator] Ubuntu Sources List Generator, https://repogen.simplylinux.ch/
• [UCS] Univention Corporate Server (UCS), https://www.univention.de/produkte/ucs/
• [VirtualBox] VirtualBox, https://www.virtualbox.org/
• [YUM] Yellowdog Updater Modified (YUM), Projektseite, http://yum.baseurl.org/
Kapitel 50

Index

- 58
--add-architecture, 32, 33
--all, 25, 213, 397, 398, 400, 404
--all-packages, 245
--anypatch, 339
--arch, 59
--audit, 227
--auto, 416
--autoclean-on-startup, 127
--batch, 370
--bug-reports, 415
--bugs, 400
--bump, 339
--bundle, 415
--cdrom add, 66
--changed, 213
--checkonly, 387
--checksig, 447
--clean-on-startup, 127
--color auto, 395
--color regular, 199
--config, 206, 213, 313
--config-file, 266
--configure, 96, 228
--contents, 10, 96, 196, 197, 339, 443
--country, 59
--debtags, 403
--default-key, 334
--depotdir, 416
--description, 339
--destdir, 440
--discard, 416
--display, 313
--display-experimental, 395
--display-format, 160
--display-info, 395
--download-only-install, 216
--empty, 266
--erase, 440
--explain, 144
--extract, 196
--field, 164, 165, 174, 187, 442
--find-config, 246
--fix-broken, 222
--fixperms, 339
--force, 244
--force-help, 250
--format, 266
--freshen, 438
--frontend, 400
--ftp, 59
--full, 329
--generate, 339
--get-selections, 312, 383
--group-by, 274
--guess, 246
--gui, 111
--help, 266, 325, 339, 416
--ignore-case, 174, 187
--ignore-default-rules, 244
--include-dist-op, 403
--include-dists, 403
--info, 133, 339, 396, 416, 446
--install, 10, 340, 341, 365, 437
--install --nodeps, 437
--install-packages, 415
--install=no, 325
--install=yes, 325
--installer, 226
--invert-match, 175, 188
--join, 416
--keep-version, 339
--libdevel, 245
--lines, 159
--list, 96, 132, 137, 172, 238, 313, 383, 406, 441–444
--list-tasks, 102
--listfiles, 10, 96, 192, 196, 197, 444
--listq, 416
--load-avail, 10
--merge-avail, 74
--msdos, 416
--multiversion, 366
--no-download-install, 413, 437
--no-empty, 266
--no-guess, 246
Debian-Paketmanagement

--no-show-section, 245
--nonfree, 59
--nopatch, 339
--npquiet, 416
--old, 404
--option, 266
--outfile, 59
--output, 416
--output=Ausgabedatei, 401
--partsize, 339
--pedantic, 395
--pending, 228
--pin-priority, 400
--print-architecture, 32, 33
--print-foreign-architectures, 32, 33
--print-uris, 414
--purge, 224, 241, 440
--purge remove, 246, 247, 440
--purge-unused, 243
--query, 313, 442
--query --configfiles, 444
--query --docfiles, 444
--query --file, 445
--query --info, 446
--query --info --package, 446
--query --list, 443, 444
--query --package, 442
--query --package --list, 443
--quiet, 244, 404, 414
--reinstall, 224, 439
--remove, 10, 237, 440
--remove-architecture, 32
--rename, 66
--requires, 442
--resolve, 440, 442
--reverse, 400
--satisfy, 101
--schedule-only, 279
--scripts, 339
--search, 10, 96, 191, 445
--severity, 399
--shell, 111
--show, 140
--show-dependents, 244
--show-depends, 244
--show-downgrade, 400
--show-field, 175, 188
--show-keepers, 244
--show-orphans, 244
--show-priority, 245
--show-section, 245
--show-size, 245
--show-summary why, 161
--showformat, 140
--sign, 333, 334
--since, 401
Debian-Paketmanagement

-Va, 448
-Vp, 448
-W, 140
-Wf, 158
-X, 132
-Z, 157
-a, 25, 59, 131, 199, 213, 228, 244–246, 400, 404, 416
-af, 196, 199
-b, 200, 370
-c, 10, 59, 86, 96, 196, 197, 213, 266, 339, 387, 443
-ce, 214
-conf, 205
-d, 167, 244, 294, 338, 416
-d add, 66
-d install, 216
-du, 157
-e, 144, 206, 213, 244, 440
-e --nodeps, 440
-f, 59, 140, 163–165, 196, 198, 200, 201, 222, 244, 252, 400, 442
-g, 339
-h, 163, 178, 266, 339
-i, 10, 174, 187, 199, 225, 244, 334, 340, 341, 365, 396, 437
-i --nodeps, 437
-ihv, 236, 437
-j, 416, 417
-k, 143, 332–334, 339
-l, 7, 10, 96, 131, 132, 137–139, 172, 178, 199, 238, 383, 416, 441–444
-ls, 196, 197
-lsbin, 200
-lsconf, 206
-lsman, 201
-m, 200, 201, 366
-man, 200
-md5check, 212, 448
-md5sum, 212, 448
-n, 59, 129–131, 159, 294
-n), 130, 131
-o, 59, 266, 342, 401, 403, 404, 416, 417
-o dir::cache::archives, 414
-p, 201, 220, 338
-q, 244, 404, 414, 442
-qR, 442
-qa, 441, 442
-qa --last, 441
-qc, 444
-qd, 444
-qf, 445
-qi, 446
-qip, 446
-ql, 443, 444
-qp, 442
-qpR, 442
-qpI, 443
-qq, 414
-qv, 243
-r, 10, 66, 170, 237, 338, 341, 440, 442
-s, 10, 59, 96, 132, 145, 155, 169, 175, 178, 188, 189, 205, 244, 245, 294, 339, 399, 416, 417, 442, 444, 446
-stdout, 178
-t, 87, 102, 318, 325, 334, 338
-t install, 237
-u, 74, 232, 370
-v, 56, 175, 188, 193, 195, 244, 266, 339, 387, 395, 401
-v why, 161
-vv, 56
-vvv, 56–59
-vvvv, 56
-x, 87, 195, 196
-z, 245
.dput.cf, 370
.mini-dinstall.conf, 369
/etc/alternatives/, 310
/etc/approx/approx.conf, 357
/etc/apt/apt.conf, 265, 268, 321, 352, 354, 423
/etc/apt/apt.conf.d/, 265
/etc/apt/apt.conf.d/70debconf, 352, 354
/etc/apt/listchanges.conf, 265
/etc/apt/preferences, 265, 269, 322
/etc/apt/preferences.d/, 265, 269, 322
/etc/apt/sources.list, 47, 48, 66, 155, 265
Änderungen vornehmen, 49
Aufbau, 48
Einträge für Deutschland, 53
Einträge für externe Ressourcen, 51
Einträge für lokale Ressourcen, 51
Einträge für nicht-offizielle Pakete, 52
Einträge für offizielle Pakete, 50
Einträge für Quellpakete, 52
Einträge für Sicherheitsaktualisierungen, 51
Einträge für Verzeichnisse, 51
Felder eines Eintrags, 49
Format, 48
Paketquelle, 48
Paketquelle nachtragen, 47, 413
Signaturen, 52
Wechsel der Veröffentlichung, 50
/etc/apt/sources.list.d/, 47, 52, 265
Paketquelle nachtragen, 47
/etc/apt/trusted.gpg, 71, 210, 265
/etc/apt/trusted.gpg.d, 210
/etc/apt/trusted.gpg.d/, 265
/etc/debian_version, 25
/etc/default/apt-cacher, 360
/etc/dpkg-www.conf, 177
/etc/dwww/dwww.conf, 177
/etc/locale.nopurge, 410
/usr/share/doc/, 256
/var/cache/apt/apt-file/, 192
/var/cache/apt/archives/, 50, 120, 121, 123, 124, 216, 225, 362
/var/cache/apt/archives/lock/, 121
/var/cache/apt/archives/partial/, 120, 121, 124, 126, 216
Debian-Paketmanagement

/var/cache/debconf, 228
/var/lib/apt/extended_states, 39
/var/lib/apt/lists, 79, 85, 192
/var/lib/aptitude/pkgstates, 39, 40
/var/lib/debfoster/keepers, 243
/var/lib/debtags/vocabulary, 288
/var/lib/dpkg/arch, 32
/var/lib/dpkg/arch-new, 32
/var/lib/dpkg/available, 140
/var/lib/dpkg/info, 38
/var/lib/dpkg/status, 39, 140, 252
/var/lib/dpkg/status-old, 253
/var/log/apt/history.log, 164
/var/log/dpkg.log, 163
Änderungen im Paketbestand, 281
Änderungen in der Paketauswahl, 43
Änderungen nachlesen, 206, 208
Änderungen vornehmen, 49
Änderungsprotokoll, 206
Überblick, 18, 35, 348
Übergangspaket, 16, 36, 90
ändern, 43, 49
öffentlich zugänglich machen, 374
über das Paketformat, 185
über den Paketinhalt, 45, 179, 180, 182
über den Paketnamen, 45, 88, 89, 172, 173, 175, 176, 179, 180, 182, 183, 185
über den Webbrowser, 55, 344
über die Paketabhängigkeiten, 185
über die Paketbeschreibung, 173–175, 179, 182, 183, 185–187
über die Paketliste, 172, 175, 179
über ein Fragment des Paketnamens, 45, 185
über ein Suchmuster, 173–176
über ein Textfragment, 173, 175, 176, 179, 180, 183
über packages.debian.org, 179
über packages.linuxmint.com, 182
über packages.tanglu.org, 181
über packages.ubuntu.com, 181
überprüfen, 209, 210, 447
~/.aptitude/config, 270
~n, 129–131

A
Abarbeitung bei der Aktualisierung, 49
Abbildung, 24
Abfolge, 254
Abgleich zwischen Paketversionen, 400
Abgrenzung zum eigenen Paketmirror, 356
Abhängigkeiten, xxvi
Abhängigkeiten anzeigen, 164, 442
Ablauf, xxvi
Abschalten der Gültigkeitsüberprüfung des Release Files
dauerhaft, 342
einmalig, 342
Acquire::Check-Valid-Until, 342
add, 66, 72, 99, 210
add-apt-repository, 47, 49, 67
Nutzung von PPAs, 67
add-user-tag, 40
Adept, 290
adept, 290
adequate, 268, 394, 397
--all, 397, 398
--tags broken-symlink, 397
--tags missing-copyright-file, 397
--tags program-name-collision, 397
Paketname, 397
administrative Benutzerrechte, xxiii
Administratoren-Umgebung, xxiii
adv, 99
Advanced Packaging Tool, 96
Aktivierung, 317
aktualisierbare Pakete anzeigen, 150, 151, 446
aktualisieren, 73, 229–232, 304, 437, 438
Aktualisierung, 7
Alias-Name, 26, 28
alien, 337, 338
--anypatch, 339
--bump, 339
--description, 339
--fixperms, 339
--generate, 339
--help, 339
--install, 341
--keep-version, 339
--nopatch, 339
--patch, 339
--scripts, 339
--single, 339
--target, 339
--test, 339
--to-deb, 338
--to-pkg, 338
--to-rpm, 338, 341
--to-slp, 338
--to-tgz, 338
--verbose, 339
--version, 339
--veryverbose, 339
-T, 339
-V, 339
-c, 339
-d, 338
-g, 339
-h, 339
-i, 341
-k, 339
-p, 338
-r, 338, 341
-s, 339
-t, 338
-v, 339
Fallen bei der Paketumwandlung, 339
Fallstricke, 339
FHS, 339
Fremdformat umwandeln und direkt installieren, 341
Paketumwandlung durchführen, 338
umgewandeltes Fremdformat installieren, 340
Voraussetzungen zur Paketumwandlung, 338
all, 4
alle Aktionen bestätigen, 283
Anbindung an lintian, 115
Android, 8
anhand der Architektur, 129, 179, 183, 185
anhand der Linux-Distribution, 185
anhand der Lizenz, 191
anhand der Paketkategorie, 45
anhand der Paketversion, 175, 179
anhand der Veröffentlichung, 45, 179, 180, 182, 183
anhand des Co-Maintainers, 190
anhand des Distributionsbereichs, 182
anhand des Maintainers, 45, 175, 179, 188–190
anhand des Uploaders, 189
anhand Regulärer Ausdrücke, 172
anhand von Deftags, 45, 294
anhand von Kurzformen (URL), 180
Anmerkungen, xxxii
Annahmen, 39
Anpassung für Ubuntu, 31
Anwendungsfall, 348
Anzahl der Einträge, 49
anzeigen, 42
apache2, 177
API, 288
apk, 8
App Centre, 119
apper, 112
appnr, 116, 431
Approx
/etc/approx/approx.conf, 357
Clientkonfiguration, 359
Installation, 357
Serverkonfiguration, 357
approx, 357
apropos, 200
APT, 11, 256–258, 263, 437
libapt-pkg, 96
Referenzliste, 256
apt, 5, 14, 36, 66, 96, 133, 134, 136, 147, 153, 185, 186, 210, 265, 266, 336, 384, 387, 400, 431
--download-only install, 216
--no-download install, 437
--reinstall, 224
-d install, 216
-f, 252
aktualisierbare Pakete anzeigen, 151
autoremove, 243
download, 215, 440
edit-sources, 49, 99
full-upgrade, 99, 230, 232, 449
install, 99, 223, 230, 336, 437, 438
install --no-download, 217
installierte Pakete aus Debian Backports anzeigen, 319
Konfiguration anzeigen, 266
list, 99, 442, 447
list --all-versions, 153
list --installed, 141, 384, 441
list --upgradable, 75, 151, 446
lokales Paket installieren, 336
Paketcache automatisch aufräumen, 127
Paketcache regelmäßig aufräumen, 127
purge, 239, 241
reinstall, 439
remove, 237, 440
safe-upgrade, 230, 231, 438
search, 99
-n, 129
-n), 130, 131
show, 99, 442
update, 47, 74, 99, 192, 230, 232, 414, 437
upgrade, 99, 232
upgrade) ((apt-get
-o dir::cache::archives, 414
APT und aptitude, 242
apt-add-repository, 67
APT-Alternativen, 11
apt-browse, 89, 183
APT-Cache, 355, 357
apt-cache, 97, 431
depends --no-breaks, 166
depends, 97, 164, 165, 244, 305, 442
depends --important, 166
depends --installed, 166
depends --no-conflicts, 166
depends --no-depends, 166, 167
depends --no-enhances, 166
depends --no-pre-depends, 166
depends --no-recommends, 166
depends --no-replaces, 166
depends --no-suggests, 166
depends --recurse, 166
depends --i, 166
dotty, 97
dump, 97
dumpavail, 97
gencaches, 97
madison, 97, 154
pkgnames, 97, 129–131, 447
policy, 33, 97, 155, 156
rdepends, 97, 169, 170, 244, 442
search, 97, 185, 441
-n, 129–131
search --full, 186
search --names-only, 173, 441
search -f, 186
search -n, 173, 441
show, 97, 133, 164, 165, 186, 446
Debian-Paketmanagement

showpkg, 8, 97, 134, 234  
showsrc, 97, 219  
stats, 97, 121  
unmet, 97, 171  
xvcg, 97  
apt-cacher, 360  
/etc/default/apt-cacher, 360  
Konfiguration des Clients, 360  
Konfiguration des Servers, 360  
apt-cacher-ng, 360  
apt-cdrom, 49, 51, 66, 97, 413  
--cdrom add, 66  
--rename, 66  
-d add, 66  
-r, 66  
add, 66  
Ident, 66  
ISO-Images, 66  
apt-cdrom-setup, 66  
apt-clone, 383, 385, 388  
clone, 385  
info, 385  
restore, 388  
restore-new-distro, 388  
apt-config, 97  
--config-file, 266  
--empty, 266  
--format, 266  
--help, 266  
--no-empty, 266  
--option, 266  
--version, 266  
-c, 266  
-h, 266  
-o, 266  
-v, 266  
dump, 123, 266  
shell, 266  
apt-dater, 393  
apt-doc, 14, 154, 256, 257  
apt-dpkg-ref, 256  
apt-dselect, 257  
apt-file, 191, 192  
-v, 193  
find, 192, 445  
list, 192, 196, 198, 443, 444  
search, 191, 192, 445  
show, 192, 193, 196, 198, 444  
Strukturdatenbank aktualisieren, 193  
update, 192, 193  
apt-get, 97, 259, 260, 431  
--download-only install, 216  
--no-download install, 413, 437  
--print-uris, 414  
--purge remove, 246, 247, 440  
--quiet, 414  
--target-release, 237  
-d install, 216  
-o, 342  
-q, 414  
-qq, 414  
t, 318  
t install, 237  
Acquire::Check-Valid-Until, 342  
autoclean, 98, 125  
autoremove, 98, 243, 254  
build-dep, 98  
changelog, 206  
check, 98, 171  
clean, 98, 125  
dist-upgrade, 98, 230, 232, 254, 449  
download, 98, 215, 413  
dselect-upgrade, 98  
install, 34, 98, 128, 222, 223, 230, 236, 302, 387, 437, 438  
install --dry-run, 221  
install --fix-broken, 236, 334, 335, 340  
install --just-print, 221  
install --no-act, 221  
install --no-download, 217  
install --recon, 221  
install --reinstall, 225, 236, 439  
install --simulate, 221  
install -f, 236, 334, 335, 340  
install -t, 305  
purge, 98, 224, 239, 241, 440  
reinstall, 439  
remove, 98, 237, 440  
remove --force-remove-essential, 37  
remove --purge, 239, 241  
source, 17, 98, 218  
update, 33, 47, 74, 79, 98, 192, 230, 232, 254, 279, 342, 414, 437  
upgrade, 98, 230–232, 254, 304, 414, 438  
upgrade --show-upgraded, 150, 446  
upgrade --simulate, 150, 446  
upgrade -s, 150  
upgrade -u, 150, 446  
upgrade -u -s, 446  
apt-get update, 253  
APT-Hooks, 270  
apt-key, 97, 210  
add, 72, 99, 210  
adv, 99  
del, 72, 99  
export, 71, 99  
exportall, 71, 99  
finger, 71, 99  
list, 71, 99  
net-update, 72, 99  
Schlüsselverwaltung, 71  
update, 72, 99  
APT-Konfiguration  
/etc/apt/apt.conf, 321
APT::Compressor, 268
APT::Default-Release, 321
APT::Install-Recommends, 268
APT::Install-Suggests, 268
APT::NeverAutoRemove, 268
APT::Periodic, 268
Dpkg::Progress-Fancy, 268

apt-listbugs, 254, 394, 399
  --bugs, 400
  --pin-priority, 400
  --severity, 399
  --show-downgrade, 400
  --stats, 399
  --tags, 399
  -B, 400
  -D, 400
  -P, 400
  -S, 399
  -T, 399
  -s, 399
apt, 400
Fehlerberichte filtern, 399, 400
Fehlerberichte sortieren, 399
list, 400
rss, 400

apt-listchanges, 265, 394, 400
  --all, 400
  --frontend, 400
  --reverse, 400
  --since, 401
  --verbose, 401
  --which, 401
  -a, 400
  -f, 400
  -v, 401
apt-mark, 97
  auto, 99
  automatic, 42
  hold, 42, 43, 137, 306
  install, 42
  manual, 42, 99
  purge, 42
  remove, 42
  showauto, 42, 99, 136, 147
  showhold, 42, 137, 306
  showinstall, 42
  showmanual, 42, 99, 136
  showpurge, 42
  showremove, 42
  unhold, 42, 43, 137, 306
apt-mirror, 374

APT-Nachfolger
  APT2, 99
  Cupt, 11
apt-offline, 412, 415
  --bug-reports, 415
  --bundle, 415

apt-offline-gui, 415
APT-Pinning
  /etc/apt/preferences, 269, 322
  /etc/apt/preferences.d, 269, 322
  Priorität eines Eintrags, 155, 269, 322
  Target Release, 321
apt-rdepends, 164, 167, 169, 170, 442
  -d, 167
  -r, 170, 442
APT-Repository, 47, 365, 368
  eigenes erstellen, 365, 368
apt-rpm, 341
apt-show-source, 220
  -p, 220
apt-show-versions, 153, 234
apt-transport-https, 14, 367
apt-utils, 14

APT-Vorgänger
  dselect, 11
apt-xapian-index, 294
APT2, 99
apt4rpm, 341
APT::Cache-Grow, 123
APT::Cache-Limit, 123
APT::Cache-Start, 123
APT::Compressor, 268
APT::Default-Release, 321
APT::Install-Recommends, 268
APT::Install-Suggests, 268
APT::NeverAutoRemove, 268
APT::Periodic, 268
aptik, 388
aptik-gtk, 388

Aptitude, 437
  Aptitude::UI::Default-Package-View, 274
  Auflistung der Pakete umsortieren, 274
  Aussehen der Textoberfläche ändern, 274
  --autoclean-on-startup, 127
  --clean-on-startup, 127
  --display-format, 160
  --download-only install, 216
  --fix-broken, 222
  --group-by, 274
  --purge-unused, 243
  --schedule-only, 279
  --show-summary why, 161
  --verbose why, 161
  -F, 160
Debian-Paketmanagement

- Installieren eines Pakets, 216
- Aktualisierung eines Pakets, 222
- Verwaltung von Vermerken, 74
- Warum, 161
Änderungen im Paketbestand, 281
add-user-tag, 40
aktualisierbare Pakete anzeigen, 151
alle Aktionen bestätigen, 283
aptitude::Theme, 276
Ausgabespalten festlegen, 272
autoclean, 125, 127
Bedienoberfläche, 12
die Ausgabe mit Platzhaltern anpassen, 271, 272
dist-upgrade, 255, 304
Dokumentation, xxx
Dokumentation offline, xxx
Dokumentation online, xxx
Dokumentationspakete, xxx
download, 104, 215, 440
forget-new, 40
Format Strings, 160, 188, 271
full-upgrade, 104, 230, 232, 255, 279, 304, 449
hold, 44
install, 104, 222, 230, 279, 283, 304, 437
install --assume-yes, 283
install --simulate, 221, 282
install --y, 283
install -s, 221, 282
installierte Pakete aus Debian Backports anzeigen, 319
interaktiver Abhängigkeitsauflöser, 105
Kategorienbrowser, 106
keep-all, 283
locale Konfigurationsdateien, 270
markauto, 43
ncurses, 12
Paketaktionen vormerken, 279, 280
Paketcache automatisch aufräumen, 126
purge, 224, 239, 241, 440
reinstall, 104, 224, 225, 439
remove, 104, 237, 279, 440
safe-upgrade, 104, 230–232, 304, 438, 449
safe-upgrade --no-new-installs, 231
versions --group-ny, 152
versions --show-package-names, 152
Vorvermerken, 279
Vorvermerken ausführen, 283
Vorvermerken simulieren, 282
why, 104, 160
why-not, 104, 162
aptitude update, 253
aptitude-common, 104
aptitude-doc, 256
aptitude-doc-en, 261
aptitude-doc-es, 31
aptitude-doc-fr, 261
aptitude-Handbuch, 261
aptitude::AutoClean-After-Update, 124
aptitude::Theme, 276
Aptitude::UI::Default-Package-View, 274
Aptosid, 451
apturl, 117, 118, 344
Ara, 290
ara, 188, 289, 290, 292
Arbeitsweise, 395
Architektur, 32
   all, 4
   architekturenabhängig, 4
   multiarch, 32
Architekturen, 2–5, 436
architekturenabhängig, 4
Armbian, 450
Asciidoc, xxviii
   auf Aktualität prüfen, 79
   auf Echtheit überprüfen, 52, 67
   auf unerwünschte Veränderungen prüfen, 209
   auf Veränderungen prüfen, 211, 447, 448
   auf Vertrauenswürdigkeit prüfen, 209, 447
Aufbau, xxxi, 47, 48, 293
Aufgaben, 18, 287
Aufgabenverteilung, 95
aufheben, 43
Auflösung von Paketabhängigkeiten, 7
Auflösung von Paketaktualisierung, 7
auflisten, 88, 89, 164, 442
Auflistung, 8
Auflistung der installierten Pakete, 7
Auflistung der Pakete umsortieren, 274
Auflistung einschränken, 166
aufräumen, 124
aufsetzen, 44, 374
ausführbare Dateien, 200
Ausfallsicherheit im Netzwerk, 356
Ausfallsicherung, 47
Ausgabe anhand von Filtern einschränken, 139
Ausgabeformate, xxviii
Ausgabespalten festlegen, 272
auskommentieren, 49
auslesen, 383
Ausnahmen, 28
Aussehen der Textoberfläche ändern, 274
Auswahl, 44, 47, 55
Auswahl anhand der Veröffentlichung, 237
Auswerten der Logdateien, 162
Auswertung, 39
Auswirkungen, 39
auto, 42, 99
autoclean, 98, 125, 127
automatic, 39, 41, 42, 147, 148, 272
automatisierte Aktualisierung, 49
Automatisierte Installation
   Preseeding, 384, 387
autoremove, 98, 243, 254
awk, 131, 414
B
Backend, 114
Backport, 31
backports.debian.org, 317
backports.org, 317
BackTrack, 451
base-files, 153, 154
Basisformat, 81
Bausteine, xxvi
Baustellenstatus, xxviii
Bedarf, 317
Bedarf an Speicherplatz, 356
Bedienoberfläche, 12
Begriff, 7, 22, 39, 120
bei Debian, 179
bei Tanglu, 181
bei Ubuntu, 181
bekannte Pakete auflisten, 129, 447
belegter Speicherplatz, 123
benennen, 328
Benennung, 25
Benennung einer Paketdatei, 30
Benutzerhandbuch zu APT, 257
Benutzerschnittstelle, 94, 96, 101, 107, 115
bereitstellen, 365, 366, 368
Beschreibung, 18, 54
Bestandteile, 85, 88, 195, 394
bestehende Installation sichern, 383
bestehende Konfiguration anzeigen, 225
bestehende Vormerkungen anzeigen, 280
bestimmte Version, 233, 236
bestimmte Version installieren, 233, 236, 237
Betriebssystem, 2, 390
Betriebssystemen, 3
Bezug eines Softwarepaketes, 355
biarch, 4
Bibliothek
   libapt-inst, 14, 93
   libapt-pkg, 12, 14, 92
   libapt-pkg-doc, 14
   libapt-pkg-perl, 14, 92
   python-apt, 92, 183
Binärpaket, 15, 50, 53, 87, 218, 324, 395
BinNMU, 31
Bo, 26
Bookworm, 26
Bordmittel, xxvii
Breite der Ausgabe festlegen, 273
Bruce Perens, 26
BSD, 452
Buch
   administrative Benutzerrechte, xxxii
   Anmerkungen, xxii
   Asciidoc, xxviii
   Aufbau, xxxi
   Ausgabeformate, xxviii
   Baustellenstatus, xxviii
   Creative Commons, xxix
   Danksagung, xxxiii
   Englischkenntnisse, xxxii
   Errata, xxxii
   Fragen, xxxii
   graphische Werkzeuge, xxxi
Debian-Paketmanagement

Debdelta-upgrade, 420
debfoster, 242, 243
  --force, 244
  --ignore-default-rules, 244
  --quiet, 244
  --show-dependents, 244
  --show-depends, 244
  --show-keepers, 244
  --show-orphans, 244
  --verbose, 244
-a, 244
-d, 244
-e, 244
-f, 244
-i, 244
-q, 244
-qv, 243
-s, 244
-v, 244
/var/lib/debfoster/keepers, 243
debget, 201, 215
debhelper, 338
Debian, 5, 71
Architekturen, 2–5, 436
architekturunabhängig, 4
Betriebssystem, 2
Betriebssystemkern, 3
Debian Archive, 28
Debian Developer, 2
Debian Free Software Guidelines (DFSG), 2, 22
Debian Quality Assurance Team, 37
Debian-Archive-Keyring, 209, 210
Debian-Gesellschaftervertrag, 22
Derivate, xxvi, 2
Distribution spiegeln, 374
Distributionsbereiche, 2
Distributionsbestandteile, 2
Dokumentation, xxvi
Einsatzbereich, 2
Einsatzzeit, 2
Entwicklungszweig, xxvi
Entwicklungszweige, xxvi, 2
Gesellschaftsvertrag, 2
Hardware, 3
Installationssmedien, 44
Installationsvarianten, 44
Keyring, 209
Lizenzen, xxvi
multiarch, 4, 5
Netzwerkinstalation, 44
Paketbezug, 44
Paketmaintainer, xxxii
Paketquelle, 44
Plattformen, 3
Plattformunterstützung, 2
Porters, 3
Ports, 3, 374, 436
Ports-Projekt, 3
Projekt, 2
Projektfinanierung, 2
Projektstruktur, 2
Projektziel, 2
Qualitätssicherung, 37
Reproducible Builds, 208
Softwareauswahl, 2
Umgang mit Aktualisierungen, 316
Varianten, 2
Veröffentlichungszyklus, xxvi
Vertrauenskette, 209
Zielgruppe, 2
Debian Administrator’s Handbook, 262
Debian Archive, 28
Debian Archive Kit (dak), 368, 374
Debian Backports, 7, 26, 316, 317, 404
Aktivierung, 317
Bedarf, 317
Dokumentation, 319
Einbindung in Debian 8 Jessie, 317
Einbindung in Debian 9 Stretch, 317
installierte Pakete anzeigen, 318, 319
Paketbezug, 318
Pakete, 317
Pflege, 317
Pinning, 318
Ubuntu, 320
Versionen, 317
Debian BTS, 18, 41, 399, 401, 402
Debian Bug Tracking System, 41, 399
Debian Developer, 2
Debian Fasttrack, 316
Debian Free Software Guidelines (DFSG), 2, 22
Debian GNU/Hurd, 452
Debian GNU/kFreeBSD, 452
Debian Live System, 451
Debian Policy Violations, 394
Debian Popularity Contest, 38
Debian Pure Blends, 117
Debian Quality Assurance Team, 37
Debian Security Team, 51, 405
Debian Snapshots
  Paketarchiv, 234
Debian Social Contract, 22
Debian Source Control, 17
Debian Sources List Generator, 49, 68
Debian-Archive-Keyring, 209, 210
debian-archive-keyring, 12, 36, 71, 210
Debian-Gesellschaftervertrag, 22
debian-goodies, 131, 158, 198–201, 215, 401
debian-handbook, 262
Debian-Installer, 226
Debian-Ports-Projekt, 71, 374
Debian-Release-Team, 28
debian-security-support, 394, 405
Debian-spezifische Schalter, 325
<table>
<thead>
<tr>
<th>Debian-Paketmanagement</th>
<th>484 / 503</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian-Spickzettel, 258</td>
<td></td>
</tr>
<tr>
<td>DebianEdu/Skolelinux, 71, 450</td>
<td></td>
</tr>
<tr>
<td>Debianpaket</td>
<td></td>
</tr>
<tr>
<td>adept, 290</td>
<td></td>
</tr>
<tr>
<td>adequate, 268, 394, 397</td>
<td></td>
</tr>
<tr>
<td>alien, 337, 338</td>
<td></td>
</tr>
<tr>
<td>apache2, 177</td>
<td></td>
</tr>
<tr>
<td>appeer, 112</td>
<td></td>
</tr>
<tr>
<td>approx, 357</td>
<td></td>
</tr>
<tr>
<td>apt, 14, 36, 66, 96, 133, 134, 136, 147, 153, 185, 186, 210, 265, 266, 336, 384, 387</td>
<td></td>
</tr>
<tr>
<td>apt-cache, 360</td>
<td></td>
</tr>
<tr>
<td>apt-cacher-ng, 360</td>
<td></td>
</tr>
<tr>
<td>apt-cdrom, 413</td>
<td></td>
</tr>
<tr>
<td>apt-cdrom-setup, 66</td>
<td></td>
</tr>
<tr>
<td>apt-clone, 383, 385, 388</td>
<td></td>
</tr>
<tr>
<td>apt-dater, 393</td>
<td></td>
</tr>
<tr>
<td>apt-doc, 14, 154, 256, 257</td>
<td></td>
</tr>
<tr>
<td>apt-dpkg-ref, 256</td>
<td></td>
</tr>
<tr>
<td>apt-file, 191, 192</td>
<td></td>
</tr>
<tr>
<td>apt-listbugs, 394, 399</td>
<td></td>
</tr>
<tr>
<td>apt-listchanges, 265, 394, 400</td>
<td></td>
</tr>
<tr>
<td>apt-mirror, 374</td>
<td></td>
</tr>
<tr>
<td>apt-offline, 412, 415</td>
<td></td>
</tr>
<tr>
<td>apt-offline-gui, 415</td>
<td></td>
</tr>
<tr>
<td>apt-rdepends, 164, 167, 169, 170</td>
<td></td>
</tr>
<tr>
<td>apt-show-source, 220</td>
<td></td>
</tr>
<tr>
<td>apt-show-versions, 153</td>
<td></td>
</tr>
<tr>
<td>apt-transport-https, 14, 367</td>
<td></td>
</tr>
<tr>
<td>apt-utils, 14</td>
<td></td>
</tr>
<tr>
<td>apt-xapian-index, 294</td>
<td></td>
</tr>
<tr>
<td>aptitude, 104, 136, 157, 187</td>
<td></td>
</tr>
<tr>
<td>aptitude-common, 104</td>
<td></td>
</tr>
<tr>
<td>aptitude-doc, 256</td>
<td></td>
</tr>
<tr>
<td>aptitude-doc-en, 261</td>
<td></td>
</tr>
<tr>
<td>aptitude-doc-es, 31</td>
<td></td>
</tr>
<tr>
<td>aptitude-doc-fr, 261</td>
<td></td>
</tr>
<tr>
<td>ara, 289, 290, 292</td>
<td></td>
</tr>
<tr>
<td>base-files, 153, 154</td>
<td></td>
</tr>
<tr>
<td>check-dfsg-status, 23, 144, 145</td>
<td></td>
</tr>
<tr>
<td>checkinstall, 324</td>
<td></td>
</tr>
<tr>
<td>command-not-found, 113, 309</td>
<td></td>
</tr>
<tr>
<td>coreutils, 12, 35, 37</td>
<td></td>
</tr>
<tr>
<td>cupt, 101</td>
<td></td>
</tr>
<tr>
<td>daptup, 76</td>
<td></td>
</tr>
<tr>
<td>dctrl-tools, 132, 155, 158, 164, 169, 174, 176, 187, 189, 199, 289, 294</td>
<td></td>
</tr>
<tr>
<td>deb-gview, 89</td>
<td></td>
</tr>
<tr>
<td>debarchiver, 377</td>
<td></td>
</tr>
<tr>
<td>debconf, 224, 225, 227, 228, 383</td>
<td></td>
</tr>
<tr>
<td>debconf-utils, 226, 383, 384, 387</td>
<td></td>
</tr>
<tr>
<td>debdelta, 420</td>
<td></td>
</tr>
<tr>
<td>debhelper, 243</td>
<td></td>
</tr>
<tr>
<td>debhelper, 338</td>
<td></td>
</tr>
<tr>
<td>debian-archive-keyring, 12, 36, 71, 210</td>
<td></td>
</tr>
<tr>
<td>debian-goodies, 131, 158, 198–201, 215, 401</td>
<td></td>
</tr>
<tr>
<td>debian-handbook, 262</td>
<td></td>
</tr>
<tr>
<td>debian-security-support, 405</td>
<td></td>
</tr>
<tr>
<td>debmirror, 376</td>
<td></td>
</tr>
<tr>
<td>deborphan, 245, 247</td>
<td></td>
</tr>
<tr>
<td>debsums, 211, 213, 448</td>
<td></td>
</tr>
<tr>
<td>debtags, 288, 289</td>
<td></td>
</tr>
<tr>
<td>debtree, 13</td>
<td></td>
</tr>
<tr>
<td>devscripts, 154, 190, 208, 219, 402</td>
<td></td>
</tr>
<tr>
<td>dh-make-perl, 324</td>
<td></td>
</tr>
<tr>
<td>diffoscope, 208, 211</td>
<td></td>
</tr>
<tr>
<td>dlocate, 132, 143, 157, 191, 200, 211, 212, 448</td>
<td></td>
</tr>
<tr>
<td>docker, 9</td>
<td></td>
</tr>
<tr>
<td>dphys-config, 328</td>
<td></td>
</tr>
<tr>
<td>dphys-swapfile, 328</td>
<td></td>
</tr>
<tr>
<td>dpkg, 35, 96, 158, 164, 165, 211, 214, 224, 227, 228, 238, 383, 416</td>
<td></td>
</tr>
<tr>
<td>dpkg-awk, 140</td>
<td></td>
</tr>
<tr>
<td>dpkg-dev, 17, 338, 366</td>
<td></td>
</tr>
<tr>
<td>dpkg-repack, 385</td>
<td></td>
</tr>
<tr>
<td>dpkg-sig, 210, 211, 333</td>
<td></td>
</tr>
<tr>
<td>dpkg-www, 89, 176, 177</td>
<td></td>
</tr>
<tr>
<td>dput, 368</td>
<td></td>
</tr>
<tr>
<td>dput-ng, 368</td>
<td></td>
</tr>
<tr>
<td>equivs, 328</td>
<td></td>
</tr>
<tr>
<td>flatpak, 9</td>
<td></td>
</tr>
<tr>
<td>ftp.debian.org, 18</td>
<td></td>
</tr>
<tr>
<td>galternatives, 310</td>
<td></td>
</tr>
<tr>
<td>gcc, 338</td>
<td></td>
</tr>
<tr>
<td>gdebi, 113, 335</td>
<td></td>
</tr>
<tr>
<td>gdebi-core, 113</td>
<td></td>
</tr>
<tr>
<td>gdebi-kde, 113</td>
<td></td>
</tr>
<tr>
<td>general, 18</td>
<td></td>
</tr>
<tr>
<td>gnome-packagekit, 112</td>
<td></td>
</tr>
<tr>
<td>gnupg, 12, 36</td>
<td></td>
</tr>
<tr>
<td>goplay, 290, 292</td>
<td></td>
</tr>
<tr>
<td>graphviz, 167</td>
<td></td>
</tr>
<tr>
<td>gtk orphan, 248</td>
<td></td>
</tr>
<tr>
<td>gzip, 163</td>
<td></td>
</tr>
<tr>
<td>how-can-i-help, 394, 404</td>
<td></td>
</tr>
<tr>
<td>htop, 134, 178</td>
<td></td>
</tr>
<tr>
<td>ia32-libs, 5</td>
<td></td>
</tr>
<tr>
<td>ice weasel, 31</td>
<td></td>
</tr>
<tr>
<td>init, 35, 37</td>
<td></td>
</tr>
<tr>
<td>isenkr am, 308</td>
<td></td>
</tr>
<tr>
<td>isenkr am-cli, 308</td>
<td></td>
</tr>
<tr>
<td>less, 35</td>
<td></td>
</tr>
<tr>
<td>libapt-inst2.0, 14, 93</td>
<td></td>
</tr>
<tr>
<td>libapt-pkg, 96</td>
<td></td>
</tr>
<tr>
<td>libapt-pkg-dev, 14</td>
<td></td>
</tr>
<tr>
<td>libapt-pkg-doc, 14, 93</td>
<td></td>
</tr>
<tr>
<td>libapt-pkg-perl, 14, 92</td>
<td></td>
</tr>
<tr>
<td>libapt-pkg5.0, 14, 92</td>
<td></td>
</tr>
<tr>
<td>libc6, 35</td>
<td></td>
</tr>
<tr>
<td>libreoffice-calc, 31</td>
<td></td>
</tr>
<tr>
<td>libreoffice-writer, 31</td>
<td></td>
</tr>
<tr>
<td>libxml2, 31</td>
<td></td>
</tr>
<tr>
<td>libxml2-utils, 31</td>
<td></td>
</tr>
<tr>
<td>libxslt1-dev, 31</td>
<td></td>
</tr>
</tbody>
</table>
Recherche anhand der Schlagworte, 289
Schlagworte, 288
Schlagworte aktualisieren, 289
Schlagworte anzeigen, 289, 291, 299
Schlagworte in aptitude anzeigen, 292
Schnittstellen, 290
Statistik, 288
Suche anhand von Schlagworten, 289–291
Vokabular, 288
debtags, 288, 289
cat, 291
diff, 300
grep, 292
grep --facets, 291, 293
grep --invert, 293
grep -i, 293
grep -names, 293
search, 292, 293
show, 291
submit, 300
tag, 291
tag add, 300
tag rm, 301
tagcat, 299
tagssearch, 299
tagshow, 300
update, 301
Debtags Browser, 296
Debtags Cloud, 296, 297
Debtags Editor, 289, 290, 297, 298
Debtags-Eintrag
   Aufbau, 293
debtags-fetch, 289, 301
debtags-submit-patch, 289, 301
debtree, 13
Deepin, 450
degrep, 199
deininstallieren, 237, 440
del, 72, 99
depends --no-breaks, 166
depends, 97, 164, 165, 244, 305, 442
depends --important, 166
depends --installed, 166
depends --no-conflicts, 166
depends --no-depends, 166, 167
depends --no-enhances, 166
depends --no-pre-depends, 166
depends --no-recommends, 166
depends --no-replaces, 166
depends --no-suggests, 166
depends --recurse, 166
depends -i, 166
deplist, 442
Derivate, xxvi, 2
des gesamten Systems überprüfen, 171
detail, 99
devscripts, 154, 190, 208, 219, 402
dfgrep, 199
DFSG, 2, 22
dg, 439
dget, 215, 219
dglob, 131
   -a, 131, 199
   -af, 196, 199
   -f, 196, 198
dgrep, 199
   --color regular, 199
   -i, 199
   -l, 199
dh-make-perl, 324
die Ausgabe mit Platzhaltern anpassen, 271, 272
die bestehende Konfiguration verwenden, 227
diff, 300
diffoscope, 208, 211
DilOS, 452
dist-upgrade, 98, 230, 232, 254, 255, 304, 449
Distribution aktualisieren, 232
   Abfolge, 254
   Release Notes, 254
   Veröffentlichungshinweise, 254
Distribution spiegeln, 374
Distributionsbereiche, 2, 18
Begriff, 22
contrib, 22
einordnung anhand der lizenzen, 22
Hintergrund der einteilung, 24
main, 22
main (Ubuntu), 23
multiverse (Ubuntu), 23
non-free, 22
non-free-firmware, 22
Paketverteilung anhand der lizenzen, 24
partner (Ubuntu), 23
restricted (Ubuntu), 23
universe (Ubuntu), 23
unterteilung bei debian, 22
unterteilung bei Ubuntu, 23
zuordnung, 22
Distributionsbestandteile, 2
Distributionswechsel
aptitude, 255
   Release Notes, 254
   Veröffentlichungshinweise, 254
Distributor, 5
dlocate, 132, 143, 157, 191, 200, 211, 212, 448
   -K, 143
   -L, 197
   -S, 191
   -conf, 205
   -du, 157
   -k, 143
   -ls, 196, 197
   -lsbin, 200
   -lsconf, 206
Debian-Paketmanagement

lsman, 201
man, 200
md5check, 212, 448
md5sum, 212, 448
-s, 132
dnf

dg, 439
downgrade, 439
erase, 440
group install, 437
group list, 447
group remove, 440
history, 448
history list all, 448
info, 442
install, 437
list, 442
list available, 447
list installed, 441, 442
list upgrades, 446
provides, 445
remove, 440
repoquery -l, 443, 444
repository-packages list --available, 447
update, 438
upgrade, 438

Docker, 9
docker, 9

Dokumentation, xxvi, xxx, 24, 263, 288, 319
/usr/share/doc/, 256
apt-get, 259
aptitude, xxx
Benutzerhandbuch zu APT, 257
dpkg, 259
Infopages, 200, 256
installierter Pakete, 201
Manpages, 256
manpages, 200
Manpages installierter Pakete anzeigen, 201
Manpages nicht installierter Pakete anzeigen, 201
nicht installierter Pakete, 201
offline, 256–258, 261–263
online, 259–263

Dokumentation offline, xxx
Dokumentation online, xxx
Dokumentationsdateien anzeigen, 444
Dokumentationsdateien auflisten, 444
Dokumentationspakete, xxx
dot, 13
dotty, 97, 167
downgrade, 7
downgrade, 439
downgrades, 233, 439
download, 98, 104, 215, 413, 440
dphys-config, 328
dphys-swapfile, 328
dpigs, 158
--lines, 159
-H, 159
-S, 159
-n, 159
--add-architecture, 32, 33
--audit, 227
--configure, 96, 228
--contents, 10, 96, 196, 197, 339, 443
--extract, 196
--field, 165, 442
--force-help, 250
--get-selections, 383
--info, 133, 339, 446
--install, 10, 340, 365
--list, 96, 132, 137, 172, 238, 383, 441, 442
--listfiles, 10, 96, 192, 196, 197, 444
--load-avail, 10
--merge-avail, 74
--pending, 228
--print-architecture, 32, 33
--print-foreign-architectures, 32, 33
--purge, 224, 241, 440
--remove, 10, 237, 440
--remove-architecture, 32
--search, 10, 96, 191, 445
--status, 10, 96, 132, 205, 442, 444, 446
--update-avail, 74
--verify, 214, 447, 448
--verify-format, 215
-C, 227
-I, 133, 446
-L, 10, 96, 192, 196, 197, 444
-P, 241, 440
-S, 10, 96, 191, 225, 445
-V, 214, 447, 448
-a, 228
-c, 10, 96, 196, 197, 339, 443
-f, 165, 442
-i, 10, 225, 334, 340, 365
-ihv, 236
-l, 7, 10, 96, 131, 132, 137–139, 172, 238, 383, 441, 442
-r, 10, 237, 440
-s, 10, 96, 132, 155, 205, 442, 444, 446
-x, 196
/var/lib/dpkg/arch-new, 32
/var/lib/dpkg/available, 140
/var/lib/dpkg/status, 140
installierte Pakete aus Debian Backports anzeigen, 318
Paketliste anhand von Filtern einschränken, 139
Referenzliste, 256
vollständige Paketliste anzeigen, 138
dpkg-architecture
Debian-Paketmanagement

dpkg-awk, 140

dpkg-deb, 10
--contents, 196, 197, 443
--extract, 196
--field, 164, 165, 442
--info, 133, 446
-l, 133, 446
-e, 88, 196, 197, 443
-f, 164, 165, 442
-x, 196

dpkg-dev, 17, 338, 366

dpkg-query, 10
--list, 132, 137
--listfiles, 196, 197, 444
--search, 191, 445
--show, 140
--showformat, 140
--status, 132, 205, 444, 446
-l, 178, 196, 197, 444
-s, 178, 191, 445
-W, 140
-Wf, 158
-f, 140
-l, 132, 137, 178
-s, 132, 205, 444, 446

dpkg-reconfigure, 224, 228

dpkg-repack, 385

dpkg-scanpackages, 366
--multiversion, 366
-m, 366

dpkg-sig, 210, 211, 333
--sign, 333
--verify, 333, 447
-k, 333

dpkg-source, 17

dpkg-split, 416
--auto, 416
--depotdir, 416
--discard, 416
--help, 416
--info, 416
--join, 416
--listq, 416
--msdos, 416
--npquiet, 416
--output, 416
--partsize, 416
--split, 416
--version, 416
?-?, 416
-l, 416
-o, 416, 417
-s, 416, 417

dpkg-www, 89, 176, 177
-h, 178
-s, 178
-stdout, 178
/etc/dpkg-www.conf, 177
/etc/dwww/dwww.conf, 177
Webserver Apache, 177
Webserver Nginx, 177

Dpkg::Progress-Fancy, 268
dput, 368
--unchecked, 370
-u, 370
dput.cf, 370
Konfiguration, 370
dput-ng, 368

Draeger OS, 452
dsc, 17
dselect, 5, 11, 257
dselect-upgrade, 98
Dummypaket, 90
dump, 97, 123, 266
dumpavail, 97
DVD, 47

Dyson OS, 452
dzgrep, 199

E

Ebenen, 95
Ebenenmodell, 10
edit-sources, 49, 99
editkeep, 242, 247
Edubuntu, 450

Effing Package Management (FPM, 437
egrep, 131
eigenes erstellen, 365, 368
Einbindung in Debian 8 Jessie, 317
Einbindung in Debian 9 Stretch, 317
einmalig, 342
Einordnung anhand der Lizenzen, 22
Einsatzbereich, 2
Einsatzzweck, 2
Einträge für Deutschland, 53
Einträge für externe Ressourcen, 51
Einträge für lokale Ressourcen, 51
Einträge für nicht-offizielle Pakete, 52
Einträge für offizielle Pakete, 50
Einträge für Quellpakete, 52
Einträge für Sicherheitsaktualisierungen, 51
Einträge für Verzeichnisse, 51
Einzelpaket aktualisieren, 304
ELTS, 29, 342, 343
emerge, 10, 260
Emmabuntüs, 450
Empfehlung für die Auswahl, 53
Englischkenntnisse, xxxii
Enrico Zini, 287, 299, 300
entfernen, 49, 237, 440
Entwicklung, 5
Entwicklungsstand, 25, 26, 320, 342
Entwicklungsziel, 2
Entwicklungszweige, xxvi, 2
Epoche, 31
equery, 10
equivs, 328
equivs-build
--full, 329
erase, 440
ergänzen, 67
erneut installieren, 224, 236, 439
erneut installieren und Konfiguration beibehalten, 224
erneut konfigurieren, 225, 228
Errata, xxxii
erstellen, 328
Erweiterte Langzeitunterstützung, 29
erzwingen, 250
essentiell, 12, 37, 200, 273
Etch, 26, 28
experimental, 25, 28
explizit entfernen, 43
explizit setzen, 43
export, 71, 99
exportall, 71, 99
Extended LTS, 29, 342, 343
extern, 47
extra, 36

F
Facetten, 288
FAI, 390
Falten bei der Paketumwandlung, 339
Fallstricke, 339
Feature Freeze, 28
Fedora
biarch, 4
Fehler melden, 18
Fehlerberichte filtern, 399, 400
Fehlerberichte sortieren, 399
Fehlerdatenbank, 399
Fehlermeldungen, 395
Felder eines Eintrags, 49
FHS, 339
file
UNIX-Kommando, 15
Filesystem Hierarchy Standard (FHS), xxxi, 338
find, 192, 445
find-file, 99
finger, 71, 99
Fingerabdruck anzeigen, 71
Fink, 452
Finnix, 451
Firmware, 24
Flatpak, 9
flatpak, 9
forget-new, 40
Forky, 26
Format, 48
Format Strings, 160, 188, 271
FPM, 437
Fragen, xxxii
Free Software Foundation (FSF), 22
freedesktop.org, 310
Freexian, 343
Freexian ELTS, 343
frei, 22
Fremdformate umwandeln und direkt installieren, 341
Fremdformate hinzufügen, 337
Frontend, 94, 96, 101, 107, 114, 115
ftp.debian.org, 18
full-upgrade, 99, 104, 230, 232, 255, 279, 304, 449

G
galternatives, 310
gcc, 338
gdebi, 113, 223, 335
Anbindung an lintian, 115
Backend, 114
Frontend, 114
gdebi-gtk, 114
gdebi-kde, 114
Paketcache aufräumen, 114
gdebi-core, 113
gdebi-gtk, 114
gdebi-kde, 113, 114
gencaches, 97
general, 18
Gesellschaftsvertrag, 2
gesicherte Konfiguration wieder einspielen, 383, 387
get, 415
gjig, 99
gnome-packagekit, 112
gnupg, 12, 36
goadmin, 290
golearn, 290
gonet, 290
gooffice, 290
goplay, 290, 292
gosafe, 290
goscience, 290
goweb, 290
gpg
--verify, 332, 333, 447
Größe festlegen, 123
Grafik, 24
graphisch darstellen, 167
graphische Werkzeuge, xxxi
Graphviz, 13
graphviz, 167
grep, 131, 204, 205, 292
grep --facets, 291, 293
grep --invert, 293
grep -i, 293
grep-aptavail, 164, 199, 289
   -F, 169
   -F Package, 169
   -P, 169
   -PX, 155
   -s, 169
grep-available, 155, 187, 289
   --field, 174, 187
   --ignore-case, 174, 187
   -F, 174, 187
   -P, 175
   -S, 175
   -i, 174, 187
grep-dctrl, 188, 289
   -F, 189
   -s, 189
grep-debtags, 289
   -d, 294
   -n, 294
   -s, 294
grep-status, 158, 164, 174, 187, 289
   --invert-match, 175, 188
   --show-field, 175, 188
   -F, 132, 169, 188, 442
   -P, 169, 442
   -X, 132
   -s, 169, 175, 188, 442
   -v, 175, 188
Grml, 451
   group install, 437
   group list, 447
   group remove, 440
   groupinstall, 437
Grundwissen, xxx
gtkorphan, 242, 248
GUI, 94, 107
gui-apt-key, 73, 454
gzip, 163

H

Hamm, 26
Hardware, 3, 390
Hardware provisioning, 390
Hierarchie, xxviii, 437
Hintergrund, 348
Hintergrund der Einteilung, 24
hinzufügen, 49
history, 448
history info, 448
history list all, 448
history package-list, 448
history packages-list, 448
history summary, 448
hold, 39, 41–44, 137, 306, 325
how-can-i-help, 394, 404
   --all, 404
   --old, 404
   --quiet, 404
   -a, 404
   -o, 404
   -q, 404
htop, 134, 178
https://screenshots.debian.net, 179
Hurd, 452

I

ia32-libs, 5
iceweasel, 31
ICMP-Pakete vs. UDP-Pakete, 59
ident, 66
Illumian, 452
important (wichtig), 36
in Anwendungen konfigurieren, 349
info, 385, 442, 446
Infopages, 200, 256
Informationen anzeigen, 71
Informationen zu Schlagworte anzeigen, 299, 300
Inhalte anzeigen, 196, 444
init, 35, 37
install, 34, 42, 98, 99, 102, 104, 107, 111, 128, 222, 223,
   230, 236, 279, 283, 302, 304, 336, 337, 367, 415, 437,
   438
install --assume-yes, 283
install --downloadonly, 440
install --dry-run, 221
install --fix-broken, 236, 334, 335, 340
install --just-print, 221
install --no-act, 221
install --no-download, 217
install --recon, 221
install --reinstall, 225, 236, 439
install --simulate, 221, 282
install --y, 283
install -f, 236, 334, 335, 340
install -s, 221, 282
install -t, 305
Installation, 357
Installation aus dem Paketcache, 217
Installationsgröße anzeigen, 157
Installationsmedien, 44
Installationsmedium einbinden, 413
Installationsvarianten, 44
installieren, 220, 365, 366, 437
installierte Pakete anzeigen, 318, 319
installierte Pakete aus Debian Backports anzeigen, 318, 319
installierter Pakete, 201
installiertes anzeigen, 137, 441
Integration in die Paketmanager, 288
Integration in tasksel, 308
interaktiver Abhängigkeitsauflöser, 105
ipkg, 8, 452
isenkram, 308
Integration in tasksel, 308
isenkram-cli, 308
ISO-Images, 66

J
Jessie, 26
jumpstart, 390

K
Kali Linux, 451
Kategoriebrowser, 106
keep-all, 283
Keryx, 412, 419
Keyring, 209
kickstart, 390
Klassifikationsschema, 288
Klonen einer bestehenden Installation, 383
Knoppix, 451
Kombination von Paketcache und eigenem Paketmirror, 356
Kommandozeile, xxxi, 94, 96
Konfiguration, 369, 370
Konfiguration ändern, 225
Konfiguration anzeigen, 266
Konfiguration bei APT, 352
Konfiguration bei Aptitude, 354
Konfiguration bei Synaptic, 352
Konfiguration des Clients, 360
Konfiguration des Servers, 360
Konfigurationsdatei, 47
/etc/alternatives/, 310
/etc/apt/apt.conf, 423
/etc/apt/sources.list, 47
/etc/apt/sources.list.d/, 47
/etc/locale.nopurge, 410
/var/cache/debconf, 228
Konfigurationsdateien anzeigen, 204, 444
Konfigurationsdateien auflisten, 204, 444
Konfigurationsdateien löschen, 246, 440
Konfigurationsmanagement, 390
konfigurieren, 225
Kubuntu, 110, 112, 450
Kurzbezeichnung, 18

L
löschen, 49, 237, 440
landscape-client, 115
landscape-client-ui, 115
landscape-client-ui-install, 115
landscape-common, 115
Langzeitunterstützung, 29, 320, 342
Leistungskriterien, 55
Lenny, 26
LernStick, 450
Lernziele, xxxii
less, 35
libapt-inst, 11, 14, 93
libapt-inst2.0, 14, 93
libapt-pkg, 11, 12, 14, 92, 96
libapt-pkg-dev, 14
libapt-pkg-doc, 14, 93
libapt-pkg-perl, 14, 92
libapt-pkg5.0, 14, 92
libc6, 35
libelektra, 430
libreoffice-calc, 31
libreoffice-writer, 31
libxml2, 31
libxml2-utils, 31
libxml2-utils, 31
LiMux, 450
lintian, 115, 394, 397, 399
 --color auto, 395
 --display-experimental, 395
 --display-info, 395
 --info, 396
 --pedantic, 395
 --verbose, 395
 -E, 395
 -I, 395
 -i, 396
 -v, 395
 Arbeitsweise, 395
 Fehlermeldungen, 395
 Tests, 395
lintian-info, 396
Linux Containers, 9
Linux Mint, 54, 182, 450
Linux Mint Debian Edition (LMDE), 450
Linux-Distribution
Debian, 5
Distributor, 5
Entwicklung, 5
Paketabhängigkeiten, 5
Paketierung, 5
Softwarestrukturen, 5
Verwaltung, 5
Linux-Zertifizierung, xxxii
list, 71, 99, 192, 196, 198, 400, 441–444, 447
list --all-versions, 153
list --available, 447
list --installed, 141, 384, 441
list --upgradable, 75, 151, 446
list all, 441
list available, 441, 447
list installed, 441, 442
list updates, 441
list updates --all, 446
list upgrades, 446
list-orphans, 99, 249
Liste der installierten Kernelpakete anzeigen, 143
listfiles, 99
Lizenz, xxix
Lizenzen, xxvi
localepurge, 410
localinstall, 341, 437
Logdatei
   /var/log/apt/history.log, 164
   /var/log/dpkg.log, 163
logrotate, 163, 164
lokal, 47, 365, 366, 368
lokal auspacken, 196
lokale Konfigurationsdateien, 270
lokales Depot, 120
lokales Paket installieren, 336
lokales Paketarchiv, 413
Lokalisierung der Ausgaben, xxxii
LPI, xxxii
   Grundwissen, xxx
   Material, xxx
   Vorbereitung, xxx
   Zertifizierung, xxx
lsb-release, 25
lsb_release
   --all, 25
   -a, 25
LTS, 29, 320, 342
LXC, 9
lxc, 9
M
madison, 97, 154
Maemo, 452
main, 22
main (Ubuntu), 23
Maintainer-Skripte, 9
   postinst, 85, 87, 222, 228, 233, 326
   postrm, 85, 87, 233
   preinst, 85, 87, 222, 233, 326
   prerem, 85, 87, 233
make, 338
Manpages, 256
manpages, 200
Manpages installierter Pakete anzeigen, 201
Manpages nicht installierter Pakete anzeigen, 201
manual, 39, 42, 99
markauto, 43
Material, xxx
Max-Age, 353
mc, 153, 328
mc-data, 328
MD5-Summen aller Dateien überprüfen, 212, 213
MD5-Summen aller Dateien anzeigen, 212
Meego, 452
mehrere Pakete überprüfen, 211
Metadaten, 15, 287
Metainformationen, 85, 195
Metapaket, 16, 90, 101, 118, 328
   benennen, 328
   erstellen, 328
   Tasks, 16
Mikro-Binärpaket, 16, 17
   udeb, 17
mini-dinstall, 368
   --batch, 370
   -b, 370
   .mini-dinstall.conf, 369
   Konfiguration, 369
minid-install, 374
minidak, 374
mintbackup, 388
mischen, 47, 324
mit alien umwandeln, 338
mit aptitude entfernen, 43
mit aptitude setzen, 43
mit wget benutzen, 350
Mitmachen, xxviii, xlix
mittels APT, 173, 185
mittels apt-browse.org, 89, 183
mittels apt-cache, 45, 173, 185
mittels apt-get.org, 183
mittels aptitude, 45, 173, 188
mittels axt-cache, 294
mittels dd-list, 190
mittels dpkg, 45, 172, 191
mittels dpkg-www, 176
mittels grep-available, 174, 187
mittels grep-dctrl, 189
mittels grep-status, 174, 187
mittels PackageKit, 45
mittels Rpmseek, 185
mittels SmartPM, 45, 176
mittels Synaptic, 45, 175, 189
mittels Webbrowser, 55
Modularität, xxvi
Multiarch, 32, 241
multiarch, 4, 5, 101, 129
multiverse (Ubuntu), 23
Muon, 5, 11, 110, 112
muon, 110, 112
Muster, 302
MX Linux, 451
N
   nach Architektur filtern, 154
   nach Binärpaketen, 180
   nach Paketen aus nicht-vertrauenswürdiger Quelle, 273
   nach Prioritäten finden, 146
   nach Veröffentlichung filtern, 154
nala, 107
   install, 107
   purge, 107
   remove, 107
   remove --purge, 107
   Namensschema, 30, 53
   Ne curses, 12
   ncurses, 12
net-update, 72, 99
netselect, 55
   -, 58
   -I, 59
   -v, 56
   -vv, 56–59
   -vvv, 56–59
   -vvvv, 56
ICMP-Pakete vs. UDP-Pakete, 59
netselect-apt, 55
   --arch, 59
   --country, 59
   --ftp, 59
   --nonfree, 59
   --outfile, 59
   --sources, 59
   -a, 59
   -c, 59
   -f, 59
   -n, 59
   -o, 59
   -s, 59
netzbasierte Installation, 53
Netzwerk, 47
Netzwerkinstallation, 44
Netzwerklast, 356
neue Pakete anzeigen, 146
Nexenta OS, 452
nginx, 38, 177
nicht aktualisieren, 306
nicht installierter Pakete, 201
nicht-offiziell, 52
NMU, 31
No-Cache, 353
No-Store, 353
non-free, 22
non-free-firmware, 22
Nummerierung, 28
nur die Paketnamen ausgeben, 140
nur herunterladen, 215, 413, 440
Nutzung von PPAs, 67
Nutzungsgrad von Paketen, 38

O
obere Ebene, 11
oci-image-tool, 9
offizielle Liste (Debian), 53
offizielle Liste (Ubuntu), 54
offline, 256–258, 261–263
ohne Paketmirror, 55
oldoldstable, 25, 28, 342
oldstable, 25, 405
online, 259–263
Online-Passung, xxviii
OpenContainers, 9
OpenMoko, 8, 112, 452
openvpn, 134, 234, 236, 237

P
OpenWrt, 8, 452
opkg, 8, 452
Optimierungspotential, 355
optional, 36
Organisation im Paketpool, 29
orphaner, 242, 247
orphans, 249
OS provisioning, 390
Pac
cap, 341
Package Installation UPgrading And Removal Testing Suite (Piuparts), 394
PackageKit, 5, 11, 112
packagekit-backend-aptcc, 113
packagekit-backend-smart, 113
packagekit-command-not-found, 113
PackageSearch, 5, 290, 292, 295
packagesearch, 290
Pacman, 10, 260
Paket
Änderungen nachlesen, 206, 208
Änderungsprotokoll, 206
Abhängigkeiten anzeigen, 164, 442
aktualisierbare Pakete anzeigen, 150, 446
aktualisieren, 229–232, 304, 438
auf unerwünschte Veränderungen prüfen, 209
auf Veränderungen prüfen, 211, 447, 448
auf Vertrauenswürdigkeit prüfen, 209, 447
bekannte Pakete auflisten, 129, 447
bereitstellen, 365, 366, 368
bestehende Konfiguration anzeigen, 225
bestimmte Version installieren, 233, 236, 237
Changelog anzeigen, 206
Dateien auf Integrität prüfen, 213
deinstallieren, 237, 440
die bestehende Konfiguration prüfen, 227
die bestehende Konfiguration verwenden, 227
Dokumentationsdaten anzeigen, 444
Dokumentationsdaten auflisten, 444
downgraden, 233, 439
Einzelpaket aktualisieren, 304
trennen, 237, 440
erneut installieren, 224, 236, 439
erneut installieren und Konfiguration beibehalten, 224
erneut konfigurieren, 225, 228
Inhalte anzeigen, 196, 444
Installation aus dem Paketcache, 217
Installationsgröße anzeigen, 157
installieren, 220, 365, 366, 437
installiertes anzeigen, 137, 441
Konfiguration ändern, 225
Konfigurationsdaten anzeigen, 204, 444
Konfigurationsdaten auflisten, 204, 444
Konfigurationsdaten löschen, 246, 440
konfigurieren, 225
löschen, 237, 440
Liste der installierten Kernelpakete anzeigen, 143
MD5-Summen aller Dateien überprüfen, 212, 213
MD5-Summen aller Dateien anzeigen, 212
mit alien umwandeln, 338
nicht aktualisieren, 306
nur herunterladen, 215, 413, 440
Paketpriorität, 35
Rückportierung, 26, 316, 317
Rückwärtsabhängigkeiten auflisten, 169, 442
Signatur überprüfen, 209, 210, 447
Signatur mehrerer Pakete überprüfen, 211
Status anzeigen, 132, 137, 441, 442, 446
Transaktionshistorie anzeigen, 448
verfügbare Pakete anzeigen, 129, 447
verfügbare Versionen anzeigen, 150, 152, 234
verifizieren, 209, 211, 447
zu Datei finden, 191, 445
Zustand anzeigen, 137, 441, 442
Zwei Paketversionen miteinander vergleichen, 208
Paket installieren, 380
über den Webbrowser, 55, 344
Auswahl anhand der Veröffentlichung, 237
Auswerten der Logdateien, 162
bestimmte Version, 233, 236
ohne Paketmirror, 55
zuletzt installierte Pakete anzeigen, 162
Paketabhängigkeiten, 5
auflisten, 164, 442
Auflistung, 8
Auflistung einschränken, 166
des gesamten Systems überprüfen, 171
graphisch darstellen, 167
Rückwärtsabhängigkeiten auflisten, 169, 442
unerfüllte Abhängigkeiten auflisten, 171
verstehen, 164
zählen, 121
Paketaktionen
Muster, 302
Paketliste, 302
vormerken, 279, 280
Paketaktionen vormerken, 279, 280
Paketarchiv, 234
Paketauswahl, 7
Paketbeschreibung, 7, 14
Basisformat, 81
Debtags, 287
Schlüsselworte für Binärpakete, 81, 325
Schlüsselworte für Sourcepakete, 83
paketbezogene Suche, 289
Paketbezug, 44, 318
Paketcache, 8, 217, 357, 362, 380
/var/cache/apt/archives/, 120, 121, 123, 124, 216, 225
/var/cache/apt/archives/lock/, 121
/var/cache/apt/archives/partial/, 120, 121, 124, 126, 216
Abgrenzung zum eigenen Paketmirror, 356
APT-Cache, 355
apt-cacher, 360
apt-cacher-ng, 360
aptitude::AutoClean-After-Update, 124
aufräumen, 124
Ausfallsicherheit im Netzwerk, 356
Bedarf an Speicherplatz, 356
Begriff, 120
belegter Speicherplatz, 123
Bezug eines Softwarepaketes, 355
Cache, 355
Dateien im Paketcache, 121
Größe festlegen, 123
Kombination von Paketcache und eigenem Paketmirror, 356
lokales Depot, 120
Netzwerklast, 356
Optimierungspotential, 355
Status anzeigen, 121
Verfallsdatum gecachter Pakete, 356
Vor- und Nachteile, 356
Zustand anzeigen, 121
Paketcache aufräumen, 114
Paketcache automatisch aufräumen, 126, 127
Paketcache regelmäßig aufräumen, 127
Pakete, 317
nach Prioritäten finden, 146
neue Pakete anzeigen, 146
Pakete aktualisieren
aktualisierbare Pakete anzeigen, 150, 446
Simulation, 150, 446
verfügbare Versionen anzeigen, 150, 152, 234
Paketflags
automatic, 147, 148
hold, 306, 325
Paketformat
apk, 8
Click-Pakete, 452
deb, 6, 8
ipkg, 8, 452
opkg, 8, 452
pkg, 8
rpm, 8
Snappy, 452
tar.gz, 8
tar.xz, 8
Paketgruppe, 16
Paketierung, 5
Paketinhalt
auflisten, 88, 89
ausführbare Dateien, 200
Bestandteile, 394
Metadaten, 15, 287
Paketintern abfragen, 93
Paketkategorie, 14
Überblick, 18
Beschreibung, 18
Kurzbezeichnung, 18
Zuordnung, 18, 287
Paketklassifikation, 287
Debian-Paketmanagement

Paketkombinationen, 16
Paketkonfiguration, 7
  auslesen, 383
    bestehende Installation sichern, 383
deconf-get-selections, 226, 383
deconf-set-selections, 227, 383
Debian-Installer, 226
gesicherte Konfiguration wieder einspielen, 383, 387
Klonen einer bestehenden Installation, 383
Paketliste, 302
  /var/lib/apt/lists/, 79
  aktualisieren, 73, 230, 437
  Ausgabe anhand von Filtern einschränken, 139
    nur die Paketnamen ausgeben, 140
    reparieren, 79
    verstehen, 137, 383
    vollständig ausgeben, 137
    vollständige Paketliste anzeigen, 138
Paketliste anhand von Filtern einschränken, 139
Paketmaintainer, xxxii, 7
  Aufgaben, 18, 287
Paketmanagement
  Auflösung von Paketabhängigkeiten, 7
  Auflösung von Paketaktualisierung, 7
  Auflistung der installierten Pakete, 7
Begriff, 7
  Paketauswahl, 7
  Paketkonfiguration, 7
  Paketmaintainer, 7
  Paketstatus, 7
Paketmarkierungen
  /var/lib/apt/extended_states, 39
  /var/lib/aptitude/pkgstates, 39, 40
  /var/lib/dpkg/status, 39
  Änderungen in der Paketauswahl, 43
  ändern, 43
  Annahmen, 39
  anzeigen, 42
  aufheben, 43
  Auswertung, 39
  Auswirkungen, 39
  auto, 42
  automatic, 39, 41, 42, 272
Begriff, 39
  Darstellung im Terminal, 41
  Darstellung in aptitude, 41
  essentiell, 12, 37, 200, 273
  explizit entfernen, 43
  explizit setzen, 43
  hold, 39, 41–43
  install, 42
  manual, 39, 42
  mit aptitude entfernen, 43
  mit aptitude setzen, 43
  purge, 42
  remove, 42
  Seiteneffekte, 43
  setzen, 42
  unhold, 43
Paketmirror, 47, 53, 355, 413
  öffentlich zugänglich machen, 374
  Abgleich zwischen Paketversionen, 400
  auf Aktualität prüfen, 79
  aufsetzen, 44, 374
  Auswahl, 44, 55
  Beschreibung, 54
  Empfehlung für die Auswahl, 53
  Leistungskriterien, 55
  Linux Mint, 54
  Namensschema, 53
  offizielle Liste (Debian), 53
  offizielle Liste (Ubuntu), 54
  Paketspiegel, 47
  primäre (Debian), 53
  sekundäre (Debian), 53, 54
  Softwareempfehlungen, 374
  spiegeln, 374
  Unterschied zu Paketcache, 374
  Varianten, 374
Paketmuster, 128
Paketname, 14, 397
  Anpassung für Ubuntu, 31
  Architektur, 32
  Backport, 31
  BinNMU, 31
  Epoche, 31
  Namensschema, 30
  NMU, 31
  Präfix, 31
  Software, 31
  Suffix, 31
  Versionsnummer, 31
  Versionsverlauf, 31
Paketoperationen
  erzwingen, 250
Paketpflege, 9
Paketpriorität, 35
  Überblick, 35
  extra, 36
  important (wichtig), 36
  optional, 36
  required (erforderlich), 35
  standard, 36
Paketprioritäten, 146
Paketproxy
  approx, 357
  apt-cacher, 360
  apt-cacher-ng, 360
Paketqualität, 37
Paketquelle, 44, 47, 48
  ändern, 49
  Abarbeitung bei der Aktualisierung, 49
  Anzahl der Einträge, 49
  APT-Repository, 47, 365, 368
auf Echtheit überprüfen, 52, 67
Aufbau, 47
Ausfallsicherung, 47
auskommentieren, 49
Auswahl, 47
automatisierte Aktualisierung, 49
CD, 47
DVD, 47
ternehmen, 49
ergänzen, 67
extern, 47
Format, 48
hinzufügen, 49
Installationsmedium einbinden, 413
Konfigurationsdatei, 47
löschen, 49
lokal, 47, 365, 366, 368
lokales Paketarchiv, 413
mischen, 47, 324
Netzwerk, 47
nicht-offiziell, 52
primäre, 47
Repository, 47
Security Updates, 51, 53
separate Einträge, 52
Sicherheitsaktualisierungen, 51, 53
Speicherkarte, 47
USB-Stick, 47
Versionskonflikte, 413
Paketquelle nachtragen, 47, 413
Paketsignatur
überprüfen, 209, 210, 447
mehrere Pakete überprüfen, 211
Paketsignaturen, 209
Auflistung, 8
Paketspiegel, 47, 374
Paketmirror, 47
Paketstatus, 7, 137, 246, 441, 442
Paketstatus anzeigen, 132, 446
Paketstatusdatenbank
/var/lib/dpkg/status, 252
/var/lib/dpkg/status-old, 253
apt-get update, 253
aptitude update, 253
Paketsuche, 44, 287, 292
über das Paketformat, 185
über den Paketinhalt, 45, 179, 180, 182
über den Paketnamen, 45, 88, 89, 172, 173, 175, 176, 179, 180, 182, 183, 185
über die Paketabhängigkeiten, 185
über die Paketbeschreibung, 173–175, 179, 182, 183, 185–187
über die Paketliste, 172, 175, 179
über ein Fragment des Paketnamens, 45, 185
über ein Suchmuster, 173–176
über ein Textfragment, 173, 175, 176, 179, 180, 183
über packages.debian.org, 179
über packages.linuxmint.com, 182
über packages.tanglu.org, 181
über packages.ubuntu.com, 181
anhand der Architektur, 129, 179, 183, 185
anhand der Linux-Distribution, 185
anhand der Lizenz, 191
anhand der Paketkategorie, 45
anhand der Paketversion, 175, 179
anhand der Veröffentlichung, 45, 179, 180, 182, 183
anhand des Co-Maintainers, 190
anhand des Distributionsbereichs, 182
anhand des Maintainers, 45, 175, 179, 188–190
anhand des Uploaders, 189
anhand Regularärer Ausdrücke, 172
anhand von Debits, 45, 294
anhand von Kurzformen (URL), 180
bei Debian, 179
bei Tanglu, 181
bei Ubuntu, 181
Linux Mint, 182
mittels APT, 173, 185
mittels apt-browse.org, 89, 183
mittels apt-cache, 45, 173, 185
mittels apt-get.org, 183
mittels aptitude, 45, 173, 188
mittels axi-cache, 294
mittels dd-list, 190
mittels dplg, 45, 172, 191
mittels dplg-www, 176
mittels grep-available, 174, 187
mittels grep-dctrl, 189
mittels grep-status, 174, 187
mittels PackageKit, 45
mittels Rpmseek, 185
mittels SmartPM, 45, 176
mittels Synaptic, 45, 175, 189
mittels Webbrowser, 55
multiarch, 129
nach Binärpaketen, 180
nach Paketen aus nicht-vertrauenswürdiger Quelle, 273
Paketumwandlung durchführen, 338
Paketransformierung
debian-archive-keyring, 12
gnupg, 12
lintian, 394
rpmlint, 394
Paketttransformation
Übergangspaket, 16, 36, 90
Binärpaket, 15, 50, 53, 87, 218, 324, 395
Dummypaket, 90
Metapaket, 16, 90, 118, 328
Mikro-Binärpaket, 16, 17
Pseudopaket, 18
Quellpaket, 52
Sourcepaket, 15, 17, 50, 52, 53, 86, 218–220, 324, 395, 406
Tasks, 16
Transitionspaket, 90
virtuelles Paket, 17, 121
Paketversion anzeigen, 153, 154
nach Architektur filtern, 154
nach Veröffentlichung filtern, 154
verfügbare Pakete, 272
Paketverwaltung
Aufgabenverteilung, 95
Benutzerschnittstelle, 94, 96, 101, 107, 115
Ebenen, 95
Frontend, 94, 96, 101, 107, 115
GUI, 94, 107
Kommandozeile, 94, 96
TUI, 94, 101
WUI, 55, 94, 115
Paketwaise, 242
Paketzustand, 137, 441, 442
partner (Ubuntu), 23
Patches, 25
PDiffs, 423
perl, 338
perl-base, 36
Perl-Modul Dpkg::Arch, 3
Pflege, 317
pforth, 33
Pinning, 318
piuparts, 394
Pixar, 26
pkg, 8
pkgnames, 97, 129–131, 447
Platte läuft voll
Paket installieren, 380
Paketcache, 380
Plattformen, 3
Plattformunterstützung, 2
Platzhalter, 131
policy, 33, 97, 155, 156
popbugs, 394, 399, 401, 402
--output=Ausgabedatei, 401
-o, 401
Popcon, 38
popcon, 401
popularity-contest, 38
Porters, 3
Ports, 3, 374, 436
Ports-Projekt, 3
postinst, 85, 87, 222, 228, 233, 326
postrm, 85, 87, 233
Potato, 26
Präfix, 31
preinst, 85, 87, 222, 233, 326
prerm, 85, 87, 233
Preseeding, 384, 387, 390
primäre, 47
primären (Debian), 53
Priorität eines Eintrags, 155, 269, 322
Prioritäten, 310
procps, 36
Projekt, 2
Projektfinanzierung, 2
Projekte, 288
Projektstruktur, 2
Projektziel, 2
provides, 445
Provisionierung, 390
Betriebssystem, 390
Hardware, 390
Rechnerklasse, 390
Proxy
Überblick, 348
Anwendungsfall, 348
Cache-Grow, 353
Cache-Limit, 353
Cache-Start, 353
Caching Proxy, 348
Hintergrund, 348
in Anwendungen konfigurieren, 349
Konfiguration bei APT, 352
Konfiguration bei Aptitude, 354
Konfiguration bei Synaptic, 352
Max-Age, 353
mit wget benutzen, 350
No-Cache, 353
No-Store, 353
SOCKS, 348
Umgebungswellen, 349
Umgebungswellen für APT, 354
unter Debian konfigurieren, 349
Varianten, 348
vor Paketmirror, 349
Proxy Server
Überblick, 348
Anwendungsfall, 348
Caching Proxy, 348
Hintergrund, 348
in Anwendungen konfigurieren, 349
Konfiguration bei APT, 352
Konfiguration bei Aptitude, 354
Konfiguration bei Synaptic, 352
mit wget benutzen, 350
SOCKS, 348
Umgebungswellen, 349
Umgebungswellen für APT, 354
unter Debian konfigurieren, 349
Varianten, 348
vor Paketmirror, 349
Pseudopakete, 18
PureOS, 450
purge, 42, 98, 107, 224, 239, 241, 440
python-apt, 92, 183
python-pkg-resources, 41
python-software-common, 67
python-software-properties, 67
python-wxversion, 419

Q
Qualitätssicherung, 37
Quellcode, xxix
Quellpaket, 52

R
Rückportierung, 26, 316, 317
Rückwärtsabhängigkeiten auflisten, 169, 442
Raspberry Pi OS, 450
Raspbian, 450
Ratemodus, 246
rc-alert, 394, 402
--debtags, 403
--include-dist-op, 403
--include-dists, 403
-d, 403
-o, 403
rc-buggy, 28
rdepends, 97, 169, 170, 244, 442
Recherche anhand der Schlagworte, 289
Rechnerklasse, 390
RedHat
   biarch, 4
Referenzierung, 25
Referenzliste, 256
reinstall, 104, 224, 225, 437, 439
Release Notes, 254
Releases, 25
remove, 42, 98, 99, 102, 104, 107, 111, 237, 279, 440
remove --force-remove-essential, 37
remove --purge, 107, 239, 241
reparieren, 79
Repo
   Paketquelle, 47
repoquery
   --list, 443, 444
   --requires, 442
   -l, 443, 444
repoquery -l, 443, 444
reportbug, 36, 399
Repository, 47
   Paketquelle, 47
Repository auf Github, xxviii, xxix
repository-packages list --available, 447
reprepro, 374
reprepro, 368
Reproducible Builds, 208
required (erforderlich), 35
restore, 388
restore-new-distro, 388
restricted (Ubuntu), 23
Rex, 26
rmadison, 154
RPM, 11, 437
   Dokumentation, xxx, 263
rpm, 8, 10, 11, 338
   --checksig, 447
   --erase, 440
   --freshen, 438
   --install, 437
   --install --nodosps, 437
   --query, 442
   --query --configfiles, 444
   --query --docfiles, 444
   --query --file, 445
   --query --info, 446
   --query --info --package, 446
   --query --list, 443, 444
   --query --package, 442
   --query --package --list, 443
   --reinstall, 439
   --requires, 442
   --upgrade, 438
   --verify, 448
   --verify --all, 448
   --verify --package, 448
   -F, 438
   -K, 447
   -U, 438
   -Va, 448
   -Vp, 448
   -e, 440
   -e --nodosps, 440
   -i, 437
   -i --nodosps, 437
   -ihv, 437
   -q, 442
   -qR, 442
   -qa, 441, 442
   -qa --last, 441
   -qc, 444
   -qd, 444
   -qf, 445
   -qi, 446
   -qip, 446
   -ql, 443, 444
   -qpl, 443
   -qpl, 443
   -qpl, 443
RPM-Paket installieren, 341
rpm2deb, 100
rpmdrake, 11
rpminstall, 100
rpmdrake, 11
rpmorphan, 242
rss, 400
rug, 260
run-parts, 265

S
safe-upgrade, 104, 230–232, 304, 438, 449
safe-upgrade --no-new-installs, 231
samba-common, 225
Sarge, 26
Schlüssel über das Netzwerk synchronisieren, 72
Schlüssel aktualisieren, 72
Schlüssel anzeigen, 71
Schlüssel exportieren, 71
Schlüssel hinzufügen, 72
Schlüssel löschen, 72
Schlüsselring
  /etc/apt/trusted.gpg, 71
  Debian, 71
  Debian-Ports-Projekt, 71
  DebianEdu/Skolelinux, 71
  Fingerabdruck anzeigen, 71
  Informationen anzeigen, 71
  Schlüssel über das Netzwerk synchronisieren, 72
  Schlüssel aktualisieren, 72
  Schlüssel anzeigen, 71
  Schlüssel exportieren, 71
  Schlüssel hinzufügen, 72
  Schlüssel löschen, 72
Ubuntu, 71
Schlüsselverwaltung, 71
Schlüsselworte für Binärpakete, 81, 325
Schlüsselworte für Sourcepakete, 83
Schlagworte, 288
Schlagworte aktualisieren, 289
Schlagworte anzeigen, 289, 291, 299
Schlagworte in aptitude anzeigen, 292
Schnittstellen, 290
screen, 180
search, 97, 99, 144, 173, 185, 187, 191, 192, 292, 293, 441, 445
  -n, 129–131
  -n), 130, 131
  ~n, 129–131
search !~akeep, 281
search --display-format, 272
search --full, 186
search --names-only, 173, 441
search --sort, 274
search --sort installsise, 159
search --width, 273
search -F, 272
search -f, 186
search -n, 173, 441
search -O, 274
search -w, 273
search ?action, 280
search ?automatic, 42, 148
search ?config-files, 239
search ?depends, 171
search ?essential, 146
search ?installed, 141, 272, 441
search ?new, 146
search ?obsolete, 149
search ?priority, 146
search ?reverse-depends, 168
search ?upgradable, 44, 151, 153, 446
search ?user-tag, 40
search -a, 280
search ~c, 239
search ~D, 171
search ~E, 146
search ~i, 104, 141, 159, 272, 441
search ~M, 42, 148
search ~m, 188
search ~N, 146
search ~o, 149
search ~p, 146
search ~R, 168
search ~U, 44, 151, 153, 446
Security Updates, 51, 53, 222, 232
sed, 131
Seiteneffekte, 43
sekundäre (Debian), 53, 54
select-editor, 310
sensible-browser, 310
sensible-editor, 310
sensible-utils, 310
separate Einträge, 52
Serverkonfiguration, 357
set, 415
setzen, 42
Shell, xxxi
shell, 266
show, 41, 97, 99, 104, 133, 136, 164, 165, 186, 192, 193, 196, 198, 291, 442, 444, 446
showauto, 42, 99, 136, 147
showhold, 42, 137, 306
showinstall, 42
showmanual, 42, 99, 136
showpkg, 8, 97, 134, 234
showpurge, 42
showremove, 42
showsrc, 97, 219
Sicherheitsaktualisierungen, 29, 51, 53, 232
Sid, 28
Siduction, 451
Signatur überprüfen, 209, 210, 447
Signatur mehrerer Pakete überprüfen, 211
Signaturen, 52
Simulation, 150, 446
skype, 23
Slackware, 8
Slax, 451
Slink, 26
smart
  --gui, 111
  --shell, 111
  install, 111
  remove, 111
  upgrade, 111
SmartPM, 5, 11, 111, 113
smartpm, 111
Snappy, 9, 452
SOCKS, 348
Software, 31
frei, 22
Patches, 25
unfrei, 22, 23
software-properties-common, 67
Softwareauswahl, 2
Softwarebibliotheken
libapt-inst, 11
libapt-pkg, 11
Softwareempfehlungen, 374
Softwareentwicklung
Abhängigkeiten, xxvi
Ablauf, xxvi
Bausteine, xxvi
Modularität, xxvi
Zerlegung in Pakete, xxvi
Softwarekomponenten, 14
Zerlegung in Pakete, 14
Softwarelizenz
Abbildung, 24
Dokumentation, 24
Firmware, 24
Grafik, 24
Softwarestapel
APT, 11
aptitude, 11
dpkg, 10
Ebenermodell, 10
Muon, 11
obere Ebene, 11
PackageKit, 11
Synaptic, 11
untere Ebene, 10
Softwarestrukturen, 5
Sortierung der Ausgabe festlegen, 274
Sortierung der Pakete nach Debtags, 106
source, 17, 98, 218
Sourcepaket, 15, 17, 50, 52, 53, 86, 218–220, 324, 395, 406
dsc, 17
Speicherkarte, 47
spiegeln, 374
Spiegelserver
öffentlich zugänglich machen, 374
aufsetzen, 374
Softwareempfehlungen, 374
Unterschied zu Paketcache, 374
Varianten, 374
Squeeze, 26
stable, 25
Stampede Linux, 338
standard, 36
Statistik, 288
stats, 97, 121
Status anzeigen, 121, 132, 137, 441, 442, 446
Steam OS, 452
StormOS, 452
Stretch, 26
Strukturdatenbank aktualisieren, 193
su, xxxii
submit, 300
Suche anhand von Schlagworten, 289–291
Suche nach obsoleten Paketen
aptitude, 149
Synaptic, 150
sudo, xxxii
Suffix, 31
Synaptic, 5, 11, 68, 107, 110, 150, 188, 189, 431
Vormerkungen, 279
synaptic, 89, 107, 177, 262, 263
sysget, 100
T
tag, 291
tag add, 300
tag rm, 301
tagcat, 299
tagsearch, 299
tagshow, 300
Tails, 451
tar.gz, 8
tar.xz, 8
Target Release, 321
Tasks, 16
taskel, 5, 16, 101
--list-tasks, 102
--task-desc, 102
--task-packages, 102
--test, 102
-t, 102
aptitude, 101
install, 102
Metapaket, 101
remove, 102
taskel-data, 101
Technische Basis, xxviii
testing, 25
Tests, 395
Tiny Core Linux, 451
Toy Story, 26
Transaktionshistorie anzeigen, 448
Transitions paket, 90
Trixie, 26
TUI, 94, 101
tzdata, 225, 228
U
ubumirror, 377
Ubuntu, 71, 320, 450
Ubuntu Landscape, 115
Ubuntu Sources List Generator, 49, 68
ubuntu-keyring, 71
Ubuntu-BSD, 452
Ubuntu-Paket
apt-clone, 385, 388
aptik, 388
apturl, 117, 118, 344
landscape-client, 115
landscape-client-ui, 115
landscape-client-ui-install, 115
landscape-common, 115
muon, 110
screen, 180
ubumirror, 377
ubuntu-keyring, 71
udeb, 17
Ultimate Debian Database (UDD), 404
Umbenennung eines Pakets, 16
Umgang mit Aktualisierungen, 316
Umgang mit Waisen
APT und aptitude, 242
UmgebungsvARIABLEN, 349
UmgebungsvARIABLEN für APT, 354
umgewandeltes Fremdformat installieren, 340
unerfüllte Abhängigkeiten auflisten, 171
unfrei, 22, 23
unhold, 42–44, 137, 306
Univention Corporate Server, 118
  App Centre, 119
  Configuration Registry, 119
Univention Corporate Server (UCS), 450
universe (Ubuntu), 23
UNIX-Kommando, 15
unmarkauto, 43
unmet, 97, 171
unreleased, 3
unstable, 3, 25, 28
unter Debian konfigurieren, 349
untere Ebene, 10
Unterschied zu Paketcache, 374
Unterteilung bei Debian, 22
Unterteilung bei Ubuntu, 23
update, 33, 47, 72, 74, 79, 98, 99, 104, 192, 193, 230, 232,
  254, 279, 301, 342, 414, 437, 438
update --security, 438
update-alternatives, 310
  --config, 313
  --display, 313
  --get-selections, 312
  --list, 313
  --query, 313
update-to, 438
updateinfo list --security, 446
updateinfo list available, 446
updateinfo list sec, 446
upgrade, 98, 99, 111, 230–232, 254, 304, 414, 438
upgrade --show-upgraded, 150, 446
upgrade --simulate, 150, 446
upgrade -s, 150
upgrade -u, 150, 446
upgrade -u -s, 446
upgrade) (),(apt-get
  -o dir::cache::archives, 414
urpmi, 11
USB-Stick, 47
util-linux, 200
V
Varianten, 2, 25, 348, 374
Veröffentlichung
  Alias-NAME, 26, 28
  Ausnahmen, 28
  Benennung, 25
  Bo, 26
  Bookworm, 26
  Bullseye, 26
  Buster, 26
  Buzz, 26
  Debian Archive, 28
  Debian Backports, 7, 26
  Debian-Release-Team, 28
  ELTS, 29, 342, 343
  Entwicklungsstand, 25, 26, 320, 342
  Erweiterte Langzeitunterstützung, 29
  Etch, 26, 28
  experimental, 25, 28
  Extended LTS, 29, 342, 343
  Feature Freeze, 28
  Forky, 26
  Freexian ELTS, 343
  Hamm, 26
  Jessie, 26
  Langzeitunterstützung, 29, 320, 342
  Lenny, 26
  LTS, 29, 320, 342
  Nummerierung, 28
  oldoldstable, 25, 28, 342
  oldstable, 25, 405
  Organisation im Paketpool, 29
  Potato, 26
  re-buggy, 28
  Referenzierung, 25
  Releases, 25
  Rex, 26
  Sarge, 26
  Sicherheitsaktualisierungen, 29
  Sid, 28
  Slink, 26
  Squeeze, 26
  stable, 25
  Stretch, 26
  testing, 25
  Toy Story, 26
  Trixie, 26
  unreleased, 3
  unstable, 3, 25, 28
Varianten, 25
Version anzeigen, 25
Versionswechsel, 28, 232
Wechsel des Entwicklungsstands, 28
Wheezy, 26, 28
Woody, 26
Veröffentlichung aktualisieren, 449
Veröffentlichungshinweise, 254
Veröffentlichungszyklus, xxvi
verfügbare Pakete, 272
verfügbare Pakete anzeigen, 129, 447
verfügbare Versionen anzeigen, 150, 152, 234
Verfallsdatum gecachter Pakete, 356
Vergleich mit DNF, xxviii, 437
Vergleich mit RPM, xxviii, 437
Vergleich mit YUM, xxviii, 437
verifizieren, 209, 211, 447
verify, 448
Version anzeigen, 25
Versionen, 317
versions, 152, 234, 441
versions --group-ny, 152
versions --show-package-names, 152
Versionskonflikte, 413
Versionsnummer, 31
Versionsverlauf, 31
Versionswechsel, 28, 232
Aktualisierung, 7
Downgrade, 7
verstehen, 137, 164, 383
Vertrauenskette, 209
Verwaltung, 5
VirtualBox, 9
virtuelles Paket, 17, 121
vnstat, 15
Vokabular, 288
vollständig ausgeben, 137
vollständige Paketliste anzeigen, 138
vor Paketmirror, 349
Vor- und Nachteile, 356
Voraussetzungen zur Paketumwandlung, 338
Vorbereitung, xxx
vormerken, 279, 280
Vormerkungen, 279
Vormerkungen ausführen, 283
Vormerkungen simulieren, 282
vrms, 23, 144, 145
--explain, 144
--sparse, 145
-e, 144
-s, 145
W
wajig, 11, 99, 242
detail, 99
find-file, 99
gijig, 99
install, 99
list-orphans, 99, 249
listfiles, 99
orphans, 249
remove, 99
rpm2deb, 100
rpminstall, 100
Webseite dazu, xxxii
Webserver Apache, 177
Webserver Nginx, 177
Wechsel der Veröffentlichung, 50
Wechsel des Entwicklungsstands, 28
Werkzeuge
Administratorrechte-Umgebung, xxxii
Bordmittel, xxvii
chroot-Umgebung, xxxii
Grundwissen, xxx
su, xxxii
sudo, xxxii
Werkzeuge zur Paketverwaltung
Hierarchie, xxviii, 437
Vergleich mit DNF, xxviii, 437
Vergleich mit RPM, xxviii, 437
Vergleich mit YUM, xxviii, 437
wget, 414
whatmaps, 268
Wheezy, 26, 28
whereis, 200
 -b, 200
 -m, 200, 201
which-pkg-broke, 215
why, 104, 160
why-not, 104, 162
Wie das Buch gepflegt wird, xxix
Wildcards, 131
wireshark, 43, 164
wireshark-common, 31
wnpp, 18
Woody, 26
WUI, 55, 94, 115
www.debian.org, 18
X
X Desktop Group, 310
x-www-browser, 310
XDG, 310
xpdf, 153
xsnow, 417
xvcg, 97
xz-utils, 12
Y
Yellowdog Updater Modified (YUM), 11, 260, 437
Yet another Setup Tool (YaST), 11
yum, 338
 check-update, 446
deplist, 442
downgrade, 439
erase, 440
groupinstall, 437
history, 448
history info, 448
history package-list, 448
history packages-list, 448
history summary, 448
info, 442, 446
install, 437
install --downloadonly, 440
list, 441
list --available, 447
list all, 441
list available, 441
list installed, 441
list updates, 441
localinstall, 341, 437
provides, 445
reinstall, 437, 439
remove, 440
update, 438
update --security, 438
update-to, 438
updateinfo list --security, 446
updateinfo list available, 446
updateinfo list sec, 446
upgrade, 438
verify, 448
yumdownloader, 440
--destdir, 440
--resolve, 440, 442

Z
zählen, 121
zcat
-f, 163
zerlegen, 416
zerlegtes Paket wieder zusammenfügen, 416
Zerlegung in Pakete, xxvi, 14
Zertifizierung, xxx
zgrep
-h, 163
Zielgruppe, xxxi, 2
zshdb, 43
zu Datei finden, 191, 445
zuletzt installierte Pakete anzeigen, 162
Zuordnung, 18, 22, 287
Zustand anzeigen, 121, 137, 441, 442
zutils, 163
Zwei Paketversionen miteinander vergleichen, 208
Zypper, 11, 260
zypper
list updates --all, 446