SelfLinux-0.13.1

APT Howto

Autor: Florian Frank (*florian@pingos.org*)
Autor: David Spreen (*netzwurm@debian.org*)
Autor: Gustavo Noronha Silva (*kov@debian.org*)
Formatierung: Florian Frank (*florian@pingos.org*)

Dieses Dokument soll dem Benutzer ein gutes Verständnis für die Arbeitsweise des Debian-Paketmanagement-Werkzeuges APT liefern. Ziel ist es, das Leben für neue Debian-Benutzer zu erleichtern und denen zu helfen, die ihr Verständnis für die Administration dieses Systems vertiefen wollen.

Inhaltsverzeichnis

1 Einführung

Am Anfang war das .tar.gz. Benutzer mussten jedes Programm, welches Sie auf ihren GNU/Linux-Systemen benutzen wollten, selbst kompilieren. Zu Beginn der Entwicklung des Debian-Projekts erachtete man es für notwendig, dass das System eine Methode zum Verwalten der Pakete, die auf dem System installiert sind, enthält. Man gab dieser Methode den Namen `dpkg`. Dadurch war das erste **Paket** auf GNU/Linux geboren, bevor Red Hat sich entschied, ihr eigenes RPM-System zu erschaffen.

Schnell standen die Macher von GNU/Linux vor einem neuen Problem. Sie brauchten ein schnelles, praktisches und effizientes Mittel, um Pakete zu installieren, das Abhängigkeiten automatisch behandeln und ihre Konfigurationsdateien während des Aktualisierens berücksichtigen würde. Und wieder war es das Debian-Projekt, das den Weg machte und APT, das **Advanced Packaging Tool**, welches seitdem von Connectiva auf RPM portiert und von einigen anderen Distributionen übernommen wurde, das Licht der Welt erblicken ließ.

Diese Anleitung versucht nicht, apt-rpm (den Connectiva-Port von APT) zu behandeln.

Diese Dokumentation basiert auf der Debian-Version: Sarge.

2 Basis-Konfiguration

2.1 Die Datei /etc/apt/sources.list

Als Teil seiner Arbeit benutzt APT eine Datei, die die **Quellen**, von denen man Pakete beziehen kann, auflistet. Diese Datei heisst /etc/apt/sources.list.

Die Einträge in dieser Datei sind von folgendem Format:

/etc/apt/sources.list

```
deb http://site.http.org/debian distribution sektion1 sektion2 sektion3
deb-src http://site.http.org/debian distribution sektion1 sektion2 sektion3
```

Natürlich sind obige Einträge erfunden und sollten nicht benutzt werden. Das erste Wort jeder Zeile, deb oder deb-src zeigt den Typ des Archivs: Entweder es enthält **Binär-Pakete** (`deb`), das sind die vorkompilierten Pakete, die wir normalerweise benutzen, oder **Quellpakete** (`deb-src`), welche die originalen Programmquellen, die Debian-Kontrolldatei (.dsc) und das diff.gz, welches die Änderungen enthält, die für das **Debianisieren** des Programms von Nöten sind.

Normalerweise finden wir folgendes in der Standard-Debian-sources.list:

/etc/apt/sources.list

```
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
```

Dieses sind die Zeilen, die eine Debian-Basis-Installation benötigt. Die erste deb-Zeile zeigt auf das offizielle Archiv, die zweite auf das Archiv non-US und die dritte auf das Archiv der Sicherheits-Updates von Debian.

Die letzten beiden Zeilen sind auskommentiert (mit einem # am Anfang). Deshalb wird apt-get sie ignorieren. Sie sind deb-src-Zeilen, das bedeutet, sie führen uns zu Debian-Quellpaketen. Wenn Sie öfters Programm-Quellen herunterladen, um sie zu testen oder neu zu kompilieren, sollten Sie die Kommentarzeichen entfernen.

Die Datei /etc/apt/sources.list kann verschiedene Typen von Zeilen enthalten. APT kann mit Archiven der Typen http, ftp und file (lokale Dateien, z. B. ein Verzeichnis, mit einem gemounteten ISO9660-Dateisystem).

Vergessen Sie nicht, `apt-get update` auszuführen, nachdem die /etc/apt-/sources.list editiert wurde. Dies ist notwendig, damit APT die Paketlisten der spezifizierten Quellen bezieht.

2.2 Wie man APT lokal benutzt

Manchmal haben Sie vielleicht einige .debs, bei denen Sie APT für die Installation benutzen wollen, so dass Abhängigkeiten automatisch aufgelöst werden.

Um das zu tun, erstellen Sie ein Verzeichnis und legen die .debs, die Sie indizieren wollen, dort hinein. Zum Beispiel:

```
root@linux / # mkdir /root/debs/
```

Es ist möglich, die Definitionen der paketeigenen Kontrolldatei (debian/control) für das Repository mit Hilfe einer override-Datei zu übergehen. In dieser Datei können Sie einige Optionen definieren, die die paketeigenen Optionen überschreiben. Das Format sieht folgendermassen aus:

```
Paket Priorität Sektion
```

Paket ist der Name des Pakets, die Priorität ist **low** (niedrig), **medium** (mittel) oder **high** (hoch) und die Sektion ist die Sektion, zu der das Paket gehört. Der Dateiname spielt keine Rolle, er muss als Argument an `dpkg-scanpackages` übergeben werden. Wenn keine override-Datei gebraucht wird, kann man `dpkg-scanpackages` auch /dev/null übergeben.

Immer noch im Verzeichnis /root führen Sie folgendes aus:

```
root@linux / # dpkg-scanpackages debs Datei | gzip > debs/Packages.gz
```

In der obenstehenden Zeile ist Datei die override-Datei. Das Kommando generiert eine Datei Packages.gz, welche verschiedene Informationen über die Pakete enthält, die APT benötigt. Um die Pakete benutzen zu können, fügen Sie folgendes der /etc/apt/sources.list hinzu:

/etc/apt/sources.list

```
deb file:/root debs/
```

Nachdem Sie das getan haben, können Sie einfach die gewöhnlichen APT-Kommandos benutzen. Sie können ebenfalls ein Quellarchiv erstellen. Die Prozedur ist dieselbe, aber die Dateien .orig.tar.gz, .dsc und .diff.gz müssen sich in dem Verzeichnis befinden und statt Packages.gz heisst es hier Sources.gz. Ausserdem müssen Sie ein anderes Programm benutzen. Es heisst `dpkg-scansources`. Das Kommando sieht folgendermassen aus:

```
root@linux / # dpkg-scansources debs | gzip > debs/Sources.gz
```

`dpkg-scansources` braucht keine override-Datei. Die Zeile in der sources.list lautet:

/etc/apt/sources.list

```
deb-src file:/root debs/
```

2.3 Entscheidung - Welcher Mirror ist der beste für die sources.list: netselect, netselect-apt

Eine häufige Frage, der meist neuen Benutzer ist: **Welchen Debian-Mirror soll ich in die sources.list eintragen?**. Es gibt viele Wege, sich für einen Mirror zu entscheiden. Die fortgeschritteneren Benutzer haben möglicherweise ein Skript, welches den Ping mehrerer Mirrors vergleicht. Aber es gibt so ein Programm inzwischen auch für weniger erfahrene Benutzer: `netselect`.

Installieren tut man `netselect` wie üblich:

```
root@linux / # apt-get install netselect
```

Wenn man es ohne Parameter ausführt, zeigt es seinen Hilfetext an. Führt man es mit einer durch Leerzeichen separierten Liste von Hostnamen (Mirrors) aus, gibt es uns einen Hostnamen zusammen mit der einer Punktzahl zurück. Diese Punktzahl berücksichtigt die erwartete Pingzeit und die Zahl der Hops (Rechner, die eine Netzwerkanfrage passiert, um ihren Zielort zu erreichen) und ist antiproportional zur erwarteten Downloadgeschwindigkeit (also je niedriger, desto besser). Angezeigt wird nur der Host mit der niedrigsten Punktzahl (Die ganze Liste der Mirrors kann mit der Option `-vv` angesehen werden). Zum Beispiel:

```
root@linux / # netselect ftp.debian.org http.us.debian.org
ftp.at.debian.org download.unesp.br ftp.debian.org.br

365 ftp.debian.org.br
```

Das bedeutet, dass von den Mirrors, die als Parameter an netselect übergeben wurden, ftp.debian.org.br der beste war mit einer Punktzahl von 365. (**Achtung!** Weil es auf meinem Computer ausgeführt wurde und die Netzwerktopographie extrem unterschiedlich und abhängig vom Standort des Computers ist, ist dieser Wert nicht notwendigerweise die richtige Geschwindigkeit für andere Computer).

Jetzt tragen Sie einfach den schnellsten Mirror in die /etc/apt/sources.list ein (sehen Sie Die Datei /etc/apt/sources.list) und befolgen Sie die Tips im Kapitel Paketverwaltung.

Hinweis: Die Liste der Mirrors ist immer auf http://www.debian.org/mirror/mirrors_full zu finden.

Ab Version 0.3 enthält das netselect-Paket das netselect-apt-Skript, das obigen Prozess automatisiert. Übergeben Sie einfach die Distribution als Parameter (Der Defaultwert ist **stable**) und die sources.list wird mit den besten **main**- und **non-US**-Mirrors generiert und im aktuellen Verzeichnis gespeichert. Das folgende Beispiel generiert eine sources.list für die stabile Distribution:

```
root@linux / # ls sources.list

ls: sources.list: File or directory not found

root@linux / # netselect-apt stable

(...)

root@linux / # ls sources.list

sources.list
```

Hinweis: Die sources.list wird im aktuellen Verzeichnis erzeugt und muss nach /etc/apt verschoben werden.

Danach befolgen Sie die Tips im Kapitel Paketverwaltung.

2.4 Hinzufügen einer CD-ROM in die sources.list

Wenn Sie lieber eine CD-ROM zum Installieren von Paketen oder Updaten ihres Systems durch APT verwenden möchten, können Sie sie in Ihre sources.list eintragen. Um dieses zu tun, können Sie das Programm apt-cdrom, wie im folgenden beschrieben, benutzen:

```
root@linux / # apt-cdrom add
```

Hierfür muss die Debian CD-ROM im Laufwerk liegen. Die CD-ROM wird gemountet, und wenn sie eine gültige Debian-CD ist, wird nach Paketinformationen gesucht. Wenn Ihre CD-ROM-Konfiguration ein wenig ungewöhnlich ist, können Sie die folgenden Optionen benutzen:

`-h`	program help
`-d directory`	CD-ROM mount point
`-r`	Rename a recognized CD-ROM
`-m`	No mounting
`-f`	Fast mode, don't check package files
`-a`	Thorough scan mode

Zum Beispiel:

```
root@linux / # apt-cdrom -d /home/kov/mycdrom add
```

Eine CD kann auch identifiziert werden ohne sie zur sources.list hinzuzufügen:

```
root@linux / # apt-cdrom ident
```

Obiges funktioniert nur, wenn das CD-ROM Laufwerk in der /etc/fstab korrekt konfiguriert ist.

3 Paketverwaltung

3.1 Update der Liste der verfügbaren Pakete

Das Paketsystem benutzt eine eigene Datenbank mit Informationen über installierte, nicht installierte und für eine Installation verfügbare Pakete. Das Programm `apt-get` benutzt diese Datenbank, um herauszufinden, wie es die vom Benutzer angeforderten Pakete installieren soll und welche zusätzlichen Pakete benötigt werden, damit die ausgewählten Pakete ordentlich funktionieren.

Um diese Liste zu updaten, benutzt man das Kommando `apt-get update`. `apt-get` sucht dann nach den Paketlisten in den Archiven aus der /etc/apt/sources.list. Im Kapitel Die Datei /etc/apt/sources.list finden Sie weitere Information über diese Datei.

Es ist eine gute Idee, dieses Kommando regelmäßig auszuführen, um sich selbst und sein System auf dem neusten Stand über mögliche Paket- bzw. Sicherheitsupdates zu halten.

3.2 Installieren von Paketen

Endlich kommt das, worauf Sie alle gewartet haben! Mit der fertigen sources.list und der Liste der verfügbaren Pakete auf dem neusten Stand ist alles, was Sie zu tun haben `apt-get` auszuführen, um das gewünschte Paket zu installieren. Zum Beispiel:

```
root@linux / # apt-get install xchat
```

APT durchsucht seine Datenbank nach der aktuellsten Version dieses Paketes und holt es aus dem entsprechenden Archiv, welches in der sources.list spezifiziert ist. Wenn es eintritt, dass das Paket von einem anderen abhängt -- wie es hier der Fall ist -- überprüft APT die Abhängigkeiten und installiert die benötigten Pakete. Sehen Sie folgendes Beispiel:

```
root@linux / # apt-get install nautilus

Reading Package Lists... Done
Building Dependency Tree... Done
  The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 0 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 17.2MB will be used.
Do you want to continue? [Y/n]
```

Das Paket `nautilus` benötigt die genannten Bibliotheken (`bonobo libmedusa0 libnautilus0`), deshalb holt APT sie aus dem Archiv. Übergibt man `apt-get` die Namen der Bibliotheken beim Aufruf mit, fragt es nicht, ob es fortfahren soll, es akzeptiert automatisch, dass die genannten Pakete installiert werden sollen.

Das bedeutet, dass APT nur um Bestätigung bittet, wenn es Pakete installieren muss, die man nicht auf der Kommandozeile übergeben hat.

Die folgenden Optionen von `apt-get` können hilfreich sein:

`-h`	Dieser Hilfetext
`-d`	Nur herunterladen - Nicht installieren oder entpacken
`-f`	Versuche fortzufahren wenn der integrity check fehlschlägt
`-s`	Nichts wirklich tun. Simulation durchführen.
`-y`	Beantworte alle Fragen mit Ja anstatt sie zu stellen.
`-u`	Zeige eine Liste der Pakete die geupgraded werden.

Es können mehrere Pakete in einer Zeile zur Installation ausgewählt werden. Pakete, die über das Netzwerk oder Internet heruntergeladen wurden, werden im Verzeichnis /var/cache/apt/archives für spätere Installationen gespeichert.

Ebenfalls kann man Pakete zum Entfernen auf derselben Zeile angeben, indem man ein -direkt hinter den Paketnamen hängt wie im folgenden:

```
root@linux / # apt-get install nautilus gnome-panel-

Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Im Abschnitt Pakete entfernen finden Sie weitere Details zum Entfernen von Paketen.

Wenn Sie ein installiertes Paket irgendwie beschädigt haben oder einfach die Dateien eines Paketes mit der aktuellsten verfügbaren Version neu installieren möchten, können Sie die Option `--reinstall` wie im folgenden nutzen:

```
root@linux / # apt-get --reinstall install gdm

Reading Package Lists... Done
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 1 reinstalled, 0 to remove and
1 not upgraded.
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

Die Version des APT, die zur Erstellung dieser Anleitung benutzt wurde, ist Version 0.5.3, die aktuelle Version in Debian **unstable** (sid) zur Zeit als sie geschrieben wurde. Wenn diese Version installiert ist, kann APT auf Ihren Wunsch noch mehr: Sie können ein Kommando der Form `apt-get install paket/distribution` benutzen, um ein Paket einer anderen Distribution zu installieren, oder `apt-get install package=version`. Zum Beispiel:

```
root@linux / # apt-get install nautilus/unstable
```

Dies installiert nautilus aus der Distribution **unstable**, auch wenn die aktuell laufende Distribution **stable** ist. Mögliche Werte für **distribution** sind stable, testing, und unstable.

Meistens ist es besser, die Option `-t` zu benutzen, um eine Distribution zu wählen, was dazu führt, dass `apt-get` diese Distribution beim Auflösen von Abhängigkeiten bevorzugt.

WICHTIG: Die **unstable**-Version von Debian ist die Version, in welcher neue Versionen von Debian-Paketen zuerst erscheinen. Diese Distribution sieht alle Änderungen, die an Paketen vorgenommen werden, kleinere und größere, welche mehrere Pakete oder das ganze System betreffen können. Aus diesem Grund sollte sie nicht von unerfahrenen Benutzern oder solchen, die geprüfte Stabilität brauchen, verwendet werden.

Die **testing**-Distribution ist ein wenig besser als **unstable** was Stabilität angeht, jedoch sollte für Produktionssysteme die Distribution **stable** benutzt werden.

3.3 Pakete entfernen

Wenn ein Paket nicht mehr gebraucht wird, kann es mit APT vom System entfernt werden. Geben Sie einfach `apt-get remove package` ein. Zum Beispiel:

```
root@linux / # apt-get remove gnome-panel

Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Wie im obigen Beispiel zu sehen ist, kümmert sich APT ebenfalls um das Entfernen der Pakete, die das Paket, das Sie entfernen wollen, benötigen. Es gibt keine Möglichkeit, Pakete mit APT zu entfernen, ohne gleichzeitig die Pakete zu entfernen, die von dem entfernten Paket abhängen.

Wenn man `apt-get` ausführt wie oben angegeben, werden die Pakete entfernt, aber die Konfigurationsdateien, falls es welche gibt, bleiben auf dem System. Für eine komplette Entfernung der Pakete, sehen Sie folgendes Beispiel:

```
root@linux / # apt-get --purge remove gnome-panel

Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Der ***** hinter den Namen der Pakete, die vom zu entfernenden Paket abhängen, bedeutet, dass deren

Konfigurationsdateien ebenso entfernt werden.

Genau wie bei der Methode `install` kann man auch bei `remove` ein Symbol benutzen, um die Wirkung für ein einzelnes Paket umzukehren. Hier fügt man einem Paket ein + zu und das Paket wird installiert, anstatt entfernt zu werden.

```
root@linux / # apt-get --purge remove gnome-panel nautilus+

Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

`apt-get` listet die Pakete auf, die extra installiert werden (die gebraucht werden, damit das Programm einwandfrei funktionieren kann), die entfernt werden und die installiert werden (hier werden die extra-Pakete noch einmal mit aufgelistet).

3.4 Upgrade von Paketen

Das Aktualisieren von Paketen ist eine tolle Sache mit APT. Es braucht dafür nur einen einzigen Befehl: `apt-get upgrade`. Man kann diesen benutzen, um Pakete aus der gleichen Distribution zu aktualisieren, oder aus einer neuen Distribution, obwohl für letzteres `apt-get dist-upgrade` empfehlenswerter ist; siehe ▶ Upgrade einer Debian-Version für weitere Einzelheiten.

Es ist sinnvoll, diesen Befehl mit der Option `-u` auszuführen. Diese Option lässt APT die komplette Liste der Pakete anzeigen, die aktualisiert werden sollen. Ohne diese Option aktualisiert man quasi blind. APT lädt die aktuellsten Versionen aller Pakete herunter und installiert sie in der richtigen Reihenfolge. Es ist wichtig, dass vor jedem Aktualisieren der Pakete `apt-get update` ausgeführt wird. Siehe Abschnitt ▶ Update der Liste der verfügbaren Pakete. Zum Beispiel:

```
root@linux / # apt-get -u upgrade

Reading Package Lists... Done
Building Dependency Tree... Done
The following packages have been kept back
  cpp gcc lilo
The following packages will be upgraded
  adduser ae apt autoconf debhelper dpkg-dev esound esound-common ftp
indent
  ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
libesd0-dev
  libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev liborbit0
  libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit procps
psmisc
29 packages upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 5055B/5055kB of archives. After unpacking 1161kB will be
used.
```

```
Do you want to continue? [Y/n]
```

Das Ganze ist extrem einfach. Die ersten paar Zeilen sagen, dass einige Pakete zurückgehalten werden (**have been kept back**). Das bedeutet, dass es neuere Versionen dieser Pakete gibt, die aus irgendeinem Grund nicht installiert werden. Mögliche Gründe sind unerfüllbare Abhängigkeiten (z.B. wenn ein Paket, von dem das neue Paket abhängt, nicht im Archiv verfügbar ist) oder neue Abhängigkeiten (das Paket hängt nun von neuen Paketen ab).

Es gibt keine **saubere** Lösung für das erste Problem. Für den zweiten Fall kann man `apt-get install` für das spezielle Paket ausführen, das zurückgehalten wurde, da dann auch die Abhängigkeiten aufgelöst werden. Eine noch sauberere Lösung ist es, `dist-upgrade` zu benutzen. Siehe Abschnitt ▶ Upgrade einer Debian-Version.

3.5 Upgrade einer Debian-Version

Diese Funktion erlaubt es, ein ganzes Debian-System entweder über das Internet oder von einer neuen CD (die sie kaufen oder aus dem Internet herunterladen können) auf einmal zu aktualisieren.

Ausserdem ist es sinnvoll, wenn an den Abhängigkeiten zwischen den Paketen Änderungen vorgenommen wurden. Mit `apt-get upgrade` werden solche Pakete nicht installiert (sie werden auf dem derzeitigen Stand gehalten **kept back**).

Wenn auf Ihrem System z.B. Revision 0 der stabilen Debian-Version läuft und Sie sich Revision 3 auf CD kaufen, können Sie APT benutzen, um ein Upgrade auf die neue Version von CD durchzuführen. Dafür benutzen Sie apt-cdrom (siehe Abschnitt ▶ Hinzufügen einer CD-ROM in die sources.list), um die CD zu ihrer /etc/apt/sources.list hinzuzufügen und führen Sie `apt-get dist-upgrade` aus.

Es ist wichtig zu wissen, dass APT immer nach der aktuellsten Version eines Pakets sucht. Wenn also Ihre /etc/apt/sources.list auf ein Archiv zeigt, das eine neuere Version eines Pakets enthält als sich auf der CD befindet, lädt APT das Paket aus diesem herunter.

In dem Beispiel aus ▶ Upgrade von Paketen sehen wir, dass manche Pakete nicht aktualisiert wurden (**kept back**). Wir werden dieses Problem nun mit der Funktion `dist-upgrade` lösen:

```
root@linux / # apt-get -u dist-upgrade

Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following NEW packages will be installed:
  cpp-2.95 cron exim gcc-2.95 libident libopenldap-runtime libopenldap1
  libpcre2 logrotate mailx
The following packages have been kept back
  lilo
The following packages will be upgraded
  adduser ae apt autoconf cpp debhelper dpkg-dev esound esound-common
ftp gcc
  indent ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
  libesd0-dev libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev
  liborbit0 libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit
  procps psmisc
31 packages upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 0B/7098kB of archives. After unpacking 3118kB will be used.
```

```
Do you want to continue? [Y/n]
```

Hier ist zu bemerken, dass die Pakete aktualisiert werden, aber neue Pakete (neue Abhängigkeiten der aktualisierten Pakete) zusätzlich installiert werden. Weiterhin wird `lilo` immer noch nicht aktualisiert, es gibt möglicherweise schwerwiegendere Probleme mit diesem Paket. Wir können das prüfen, in dem wir folgendes ausführen:

```
root@linux / # apt-get -u install lilo

Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be REMOVED:
  debconf-tiny
The following NEW packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be upgraded
  lilo
1 packages upgraded, 9 newly installed, 1 to remove and 31 not upgraded.
Need to get 225kB/1179kB of archives. After unpacking 2659kB will be
used.
Do you want to continue? [Y/n]
```

Hier erfahren wir, dass das neue `lilo` einen Konflikt mit dem Paket `debconf-tiny` hat, was bedeutet, dass wir es nicht installieren (oder aktualisieren) können, ohne `debconf-tiny` zu entfernen.

Um herauszufinden, wovon ein Paket zurückgehalten oder entfernt wird, können Sie folgendes tun:

```
root@linux / # apt-get -o Debug::pkgProblemResolver=yes dist-upgrade

Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Starting
Starting 2
Investigating python1.5
Package python1.5 has broken dep on python1.5-base
  Considering python1.5-base 0 as a solution to python1.5 0
  Holding Back python1.5 rather than change python1.5-base
Investigating python1.5-dev
Package python1.5-dev has broken dep on python1.5
  Considering python1.5 0 as a solution to python1.5-dev 0
  Holding Back python1.5-dev rather than change python1.5
 Try to Re-Instate python1.5-dev
Done
Done
The following packages have been kept back
  gs python1.5-dev
0 packages upgraded, 0 newly installed, 0 to remove and 2  not upgraded.
```

Auf diesem Wege ist es einfach festzustellen, dass das Packet python1.5-dev wegen einer ungelösten Abhängigkeit zu python1.5 nicht installiert werden kann.

3.6 Ungenutzte Pakete entfernen: apt-get clean and autoclean

Wenn ein Paket installiert werden soll, bezieht APT von den Quellen, die in der /etc/apt/sources.list aufgelistet sind, die nötigen Dateien, legt sie in ein lokales Archiv (/var/cache/apt/archives/) und fährt mit der Installation fort. (sehen Sie ▶ Installieren von Paketen).

Nach und nach kann dieses lokale Archiv immer größer werden und eine Menge Platz auf der Festplatte belegen. Auch für diesen Fall bietet APT Werkzeuge an, um sein lokales Archiv zu warten: apt-get clean und autoclean Methoden.

apt-get clean entfernt alles bis auf Lock-Dateien aus /var/cache/apt/archives/ und /var/cache/apt/archives/partial/. In der Folge muss APT ein Paket, das Sie erneut installieren wollen auch erneut herunterladen.

apt-get autoclean entfernt nur Pakete, die nicht mehr heruntergeladen werden können.

Das folgende Beispiel sollte zeigen, wie apt-get autoclean arbeitet:

```
root@linux / # ls /var/cache/apt/archives/logrotate*
/var/cache/apt/archives/gpm*

logrotate_3.5.9-7_i386.deb
logrotate_3.5.9-8_i386.deb
gpm_1.19.6-11_i386.deb
```

In /var/cache/apt/archives liegen zwei Versionen des Pakets logrotate und eine Version des Pakets gpm.

```
root@linux / # apt-show-versions -p logrotate

logrotate/stable uptodate 3.5.9-8

root@linux / # apt-show-versions -p gpm

gpm/stable upgradeable from 1.19.6-11 to 1.19.6-12
```

apt-show-versions zeigt, dass logrotate_3.5.9-8_i386.deb die aktuelle Version von logrotate bereitstellt, daher wird logrotate_3.5.9-7_i386.deb nicht mehr benötigt. Ebenso wird gpm_1.19.6-11_i386.deb nicht mehr benötigt, da eine aktuellere Version von den Debian-Archiven heruntergeladen werden kann.

```
root@linux / # apt-get autoclean

Reading Package Lists... Done
Building Dependency Tree... Done
Del gpm 1.19.6-11 [145kB]
Del logrotate 3.5.9-7 [26.5kB]
```

`apt-get autoclean` entfernt also nur die alten Pakete. Für weitere Informationen über `apt-show-versions` sehen Sie Upgrade von Paketen spezieller Debian-Versionen.

3.7 APT unter dselect verwenden...

`dselect` ist ein Programm, das Debian-Benutzern hilft, zu installierende Pakete auszuwählen. Viele halten es für zu kompliziert und vielmehr langweilig, aber mit ein wenig Übung kann man durchaus Gefallen an seiner konsolen-basierten ncurses-Oberfläche finden.

Eine Stärke von `dselect` ist es, dass es mit den Möglichkeiten umgehen kann, die Debian-Pakete haben, um andere Pakete zu empfehlen (**suggesting**) oder vorzustellen (**recommending**). Um es zu benutzen, rufen Sie `dselect` als **root** auf. Wählen Sie **apt** als Zugriffsmethode. Das ist zwar nicht dringend notwendig, aber wenn Sie keine CD-ROM benutzen und Pakete aus dem Internet herunterladen möchten, ist es der beste Weg.

Um ein besseres Verständnis über die Benutzung von `dselect` zu erhalten, lesen sie die Dokumentation auf der Debian-Homepage http://www.debian.org/doc/ddp.

Nachdem Sie Ihre Auswahl mit `dselect` getroffen haben, benutzen Sie folgendes Kommando

```
root@linux / # apt-get -u dselect-upgrade
```

wie im folgenden Beispiel:

```
root@linux / # apt-get -u dselect-upgrade

Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  lbxproxy
The following NEW packages will be installed:
  bonobo console-tools-libs cpp-3.0 enscript expat fingerd gcc-3.0
  gcc-3.0-base icepref klogd libdigest-md5-perl libfnlib0 libft-perl
  libgc5-dev libgcc300 libhtml-clean-perl libltdl0-dev libsasl-modules
  libstdc++3.0 metamail nethack proftpd-doc psfontmgr python-newt talk
tidy
  util-linux-locales vacation xbill xplanet-images
The following packages will be upgraded
  debian-policy
1 packages upgraded, 30 newly installed, 1 to remove and 0  not upgraded.
Need to get 7140kB of archives. After unpacking 16.3MB will be used.
Do you want to continue? [Y/n]
```

Im Vergleich: `apt-get dist-upgrade` auf demselben System:

```
root@linux / # apt-get -u dist-upgrade

Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be upgraded
  debian-policy
1 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
```

```
Need to get 421kB of archives. After unpacking 25.6kB will be freed.
Do you want to continue? [Y/n]
```

Viele der Pakete im oberen Beispiel werden installiert, weil andere Pakete sie **empfehlen** (**suggest** or **recommend**). Andere werden aufgrund unserer Auswahl, die wir beim Navigieren durch die Paketlisten von `dselect` getroffen haben, installiert oder entfernt (Im Falle von `lbxproxy` z. B.). `dselect` kann in Verbindung mit `APT` ein nützliches Werkzeug sein.

3.8 Wartung eines "gemischten" Systems

Viele benutzen die **testing**-Distribution, da sie stabiler ist als **unstable** und aktueller als **stable** ist. Benutzer, die aktuelle Versionen von Paketen wollen, sich aber nicht trauen, ihr ganzes System auf **unstable** umzustellen, haben die Möglichkeit **testing** und **unstable** zu mischen. Auf der anderen Seite möchten konservativere Benutzer vielleicht **stable** und **testing** mischen.

Für diesen Zweck muss die folgende Zeile in die `/etc/apt/apt.conf` eingefügt werden:

/etc/apt/apt.conf

```
APT::Default-Release "testing";
```

Um nun Pakete aus unstable zu installieren, muss die Option `-t` benutzt werden:

```
root@linux / # apt-get -t unstable install Paketname
```

Vergessen Sie aber nicht, dass sie, um ein Paket aus einer anderen Debian Version zu installieren, eine Quellzeile in die /etc/apt/sources.list einfügen müssen. In unserem Beispiel brauchen wir Quellzeilen für die **unstable**-Distribution neben denen für **testing**.

3.9 Upgrade von Paketen spezieller Debian-Versionen

`apt-show-versions` bietet einen sicheren Weg für Benutzer gemischter Systeme, um ihre Systeme zu aktualisieren, ohne mehr aus der weniger stabilen Distribution zu installieren als sie im Sinn haben. Zum Beispiel ist es möglich, nur die **unstable**-Pakete zu aktualisieren, in dem man folgendes ausführt:

```
root@linux / # apt-get install `apt-show-versions -u -b | grep unstable`
```

3.10 Wie man bestimmte Versionen eines Paketes behält (komplex)

Manchmal gibt es Gründe, etwas in einem Paket zu verändern, und es fehlt die Zeit oder die Lust, diese Dinge auf neue Versionen des Paketes zu übertragen. Vielleicht haben Sie auch gerade ihre Debian-Version auf 3.0 aktualisiert, aber möchten trotzdem ein Paket aus Version 2.2 behalten. Es ist möglich, die installierte Version zu markieren (**pin**), so dass sie nicht aktualisiert wird.

Diese Möglichkeit einzusetzen ist einfach. Editieren Sie einfach die Datei /etc/apt/preferences.

Das Format ist trivial:

/etc/apt/preferences

```
Package: <Paket>
Pin: <Pin-Definition>
Pin-Priority: <Priorität des Pins>
```

Um zum Beispiel das Paket sylpheed in Version **0.4.99** zu behalten, fügen wir folgendes hinzu:

/etc/apt/preferences

```
Package: sylpheed
Pin: version 0.4.99*
```

Beachten Sie das ***** (Sternchen/Asterisk). Es funktioniert als Platzhalter; das bedeutet, dass dieser **Pin** für alle Versionen, die mit **0.4.99** beginnen, gültig sein soll. Das ist nötig, da Pakete in Debian eine Nummer für die **Debian-Revision** enthalten und ich nicht verhindern möchte, dass diese Revisionen installiert werden. Folglich werden die Versionen **0.4.99-1** und **0.4.99-10** installiert, sobald sie verfügbar sind. Beachten Sie, dass Sie das vermutlich nicht möchten, wenn Sie das Paket modifiziert haben, da diese Änderungen dann verloren gehen.

Das Feld Pin-Priority ist optional; wenn nicht anders spezifiziert, hat der **Pin** die Priorität 989.

Lassen Sie uns einen Blick darauf werfen, wie die Pin-Priorität funktioniert. Eine niedrigere Priorität als Null bewirkt, dass das Paket nie installiert wird. Prioritäten von 0 bis 100 bezeichnen Pakete, die nicht installiert sind und keine verfügbare Version haben. Solche Pakete werden in der Auswahl verfügbarer Versionen nicht berücksichtigt. Die Priorität 100 bezeichnet ein installiertes Paket. Damit eine installierte Version von einer anderen ersetzt wird, muss die Priorität über 100 liegen.

Prioritäten über 100 sagen aus, dass ein Paket installiert werden soll. Normalerweise wird ein installiertes Paket nur durch neuere Versionen ersetzt. Alle Prioritäten zwischen 100 und 1000 (inklusive) führen zu diesem Normalverhalten. Ein Paket mit solcher Priorität wird nicht durch eine niedrigere Version ersetzt. Wenn ich also zum Beispiel sylpheed 0.5.3 installiert habe und einen Pin auf sylpheed 0.4.99 mit der Priorität 999 definiert habe, wird Version 0.4.99 nicht installiert werden, um den Pin zu erfüllen. Um Pakete deaktualisieren zu können, um einen Pin zu erfüllen, braucht der Pin eine Priorität von über 1000.

Ein Pin kann für die Version, das Release oder die Herkunft (origin) definiert werden.

Für einen Pin auf die Version, wie oben beschrieben, kann man sowohl Versionsnummern als auch Platzhalter (Sternchen) verwenden. Letzteres spezifiziert mehrere Versionen in einem Pin.

Die Option Release benutzt die Release-Datei aus den APT-Archiven oder von den CDs. Die Brauchbarkeit dieser Version verfällt, wenn Sie APT-Archive benutzen, die diese Datei nicht zur Verfügung stellen. Sie können den Inhalt der Release-Dateien, die Sie haben in /var/lib/apt/lists/ nachlesen. Die Parameter für ein Release sind: a (Archiv(Archive)), c (Sektion(Component)), v (Version(Version)), o (Herkunft(Origin)) und l (Label(Label)).

Beispiel:

/etc/apt/preferences

```
Package: *
Pin: release v=2.2*,a=stable,c=main,o=Debian,l=Debian
Pin-Priority: 1001
```

In diesem Beispiel wählen wir die Debian-Version 2.2* (was 2.2r2, 2.2r3 sein kann -- **r*** bezeichnet so genannte **point releases**, welche normalerweise Sicherheitsupdates und andere extrem wichtige Updates enthalten), das stable Archiv, die Sektion main (im Gegensatz zu contrib oder non-free) und Herkunft und Label Debian. Herkunft (o=) bezeichnet, wer die Release-Datei erstellt hat, das Label (l=) definiert den Namen der Debian-Distribution: Debian für Debian selbst und Progeny für Progeny Linux zum Beispiel. Ein Beispiel einer Release-Datei:

```
root@linux / # cat
/var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-i386_Release

Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

4 Sehr nützliche Helfer

4.1 Installieren selbstkompilierter Pakete: equivs

Manchmal will man spezielle Versionen eines Programms benutzen, die nur als Quellcode verfügbar sind und nicht als Debian-Paket. Hier kann es allerdings Probleme mit dem Paket-System geben. Angenommen Sie wollen eine neue Version ihres Mailservers kompilieren und alles klappt, aber viele Pakete in Debian hängen von einem MTA (Mail Transfer Agent) ab. Da etwas installiert wurde, was Sie selbst kompiliert haben, weiss das Paketsystem darüber nicht Bescheid.

Hier kommt das `equivs` ins Spiel. Um es zu benutzen, installieren Sie das Paket mit diesem Namen. Es erstellt ein leeres Paket, das die Abhängigkeiten erfüllt und dem Paketsystem mitteilt, so dass es keine Probleme mit Abhängigkeiten gibt.

Bevor wir näher darauf eingehen, ist es wichtig, Sie darauf hinzuweisen, dass es sicherere Möglichkeiten gibt, Programme, für die in Debian schon Pakete existieren, mit anderen Optionen zu kompilieren und man equivs nicht benutzen sollte, um Abhängigkeiten zu entfernen, ohne genau zu wissen, was man tut. Siehe Das Arbeiten mit Quellpaketen für nähere Informationen.

Lassen Sie uns mit dem MTA-Beispiel fortfahren. Sie haben also gerade ihren frisch kompilierten `postfix` installiert und wollen nun mutt (ein Mailprogramm) installieren. Plötzlich stellen Sie fest, dass mutt einen anderen MTA installieren möchte, obwohl Sie schon ihren selbstkompilierten MTA laufen haben.

Wechseln Sie in irgendein Verzeichnis (z. B. /tmp), und führen Sie folgendes aus:

```
root@linux / # equivs-control name
```

Ersetzen Sie **name** durch den Namen der Kontrolldatei, die Sie erstellen wollen. Die Datei wird wie folgt erstellt:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: <Paketname; wenn nicht angegeben: equivs-dummy>
Version: <Versionsnummer; wenn nicht angegeben: 1.0>
Maintainer: <Ihr Name mit Emailadresse; wenn nicht angegeben: Benutzername>
Pre-Depends: <Pakete>
Depends: <Pakete>
Recommends: <Pakete>
Suggests: <Pakete>
Provides: <(virtuelles) Paket>
Architecture: all
Copyright: <copyright Datei; normalerweise GPL2>
Changelog: <changelog file; normalerweise ein generisches Changelog>
Readme: <README.Debian file; wenn nicht angegeben, ebenfalls ein generisches>
Extra-Files: <Zusätzliche Dateien für das doc-Verzeichnis, kommasepariert>
Description: <kurze Beschreibung; Standard ist "some wise words">
 Lange Beschreibung und Info
 .
 Zweiter Paragraph
```

Nun muss das so angepasst werden, dass es tut, was wir wollen. Schauen Sie sich die Felder und ihre Beschreibungen an, es ist nicht nötig, jedes einzelne hier zu erklären, lassen Sie uns das Nötigste tun:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: mta-local
Provides: mail-transport-agent
```

Das war es schon. mutt hängt von mail-transport-agent ab, was ein virtuelles Paket ist, das alle MTAs liefern. Ich hätte das Paket einfach mail-transport-agent nennen können, aber ich bevorzugte das Schema für virtuelle Pakete, welches das Feld **Provides** benutzt.

Nun muss das Paket nur noch gebaut werden:

```
root@linux / # equivs-build name

dh_testdir
touch build-stamp
dh_testdir
dh_testroot
dh_clean -k
# Add here commands to install the package into debian/tmp.
touch install-stamp
dh_testdir
dh_testroot
dh_installdocs
dh_installchangelogs
dh_compress
dh_fixperms
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: building package `name' in `../name_1.0_all.deb'.

The package has been created.
Attention, the package has been created in the current directory,
```

Und nun installieren Sie das erzeugte .deb.

Wie man unschwer erkennen kann, gibt es verschiedene Anwendungen für `equivs`. Man könnte sogar ein Favoriten-Paket erstellen, was von den Paketen abhängt, die Sie normalerweise installieren. Lassen Sie Ihren Vorstellungen einfach freien Lauf, aber seien Sie **vorsichtig**.

Es ist wichtig zu erwähnen, dass es in /usr/share/doc/equivs/examples einige Beispiel-Kontrolldateien gibt. Werfen Sie mal einen Blick darauf.

4.2 Entfernen von unbenutzten locale-Dateien: localepurge

Viele Debian-Benutzer verwenden nur ein **locale** (Spracheinstellung). Ein brasilianischer Debian-Benutzer benutzt z. B. vermutlich immer das brasilianische pt_BR-locale und interessiert sich nicht für das spanische es-locale.

`localepurge` ist ein sehr nützliches Werkzeug für diese Art von Benutzern. Sie können eine Menge Festplattenplatz sparen, wenn Sie nur die locales installiert haben, die Sie auch wirklich brauchen. Installieren Sie einfach `apt-get install localepurge`.

Es ist wirklich einfach zu konfigurieren, `Debconf`-Fragen führen den Benutzer Schritt für Schritt durch die Konfiguration. Seien Sie vorsichtig beim Beantworten der ersten Frage, da falsche Antworten alle **locales** entfernen können - selbst die, die Sie benutzen. Die einzige Möglichkeit, sie wiederherzustellen, ist, alle Pakete neu zu installieren, die sie enthalten.

4.3 Erfahren, welche Pakete aktualisiert werden können

`apt-show-versions` ist ein Programm, das zeigt, welche Pakete im System aktualisiert werden können und andere hilfreiche Informationen bietet. Die Option `-u` zeigt eine Liste der Pakete, die aktualisiert werden können:

```
root@linux / # apt-show-versions -u

libeel0/unstable upgradeable from 1.0.2-5 to 1.0.2-7
libeel-data/unstable upgradeable from 1.0.2-5 to 1.0.2-7
```

5 Informationen über Pakete

Es gibt einige Oberflächen für das APT-System, die es signifikant einfacher machen, Listen über verfügbare Pakete oder schon installierte Pakete zu bekommen oder auch herauszufinden, zu welcher Sektion ein Paket gehört, welche Priorität es hat, wie seine Beschreibung lautet, etc.

Unser Ziel aber hier ist, APT selbst benutzen zu lernen. Wie können wir also den Namen eines Paketes herausfinden, welches wir installieren wollen?

Es gibt eine Reihe von Möglichkeiten für eine solche Aufgabe. Fangen wir mit apt-cache an. Dieses Programm wird vom APT-System zum Warten seiner Datenbank benutzt. Werfen wir nur einen kleinen Blick auf einige seiner praktischeren Anwendungen.

5.1 Paketnamen entdecken

Angenommen Sie wollen die alten Zeiten des Atari 2600 wieder aufleben lassen. Sie möchten APT benutzen, um einen Atari-Emulator zu installieren und dann Spiele herunterladen. Sie haben folgende Möglichkeit:

```
root@linux / # apt-cache search atari

atari-fdisk-cross - Partition editor for Atari (running on non-Atari)
circuslinux - The clowns are trying to pop balloons to score points!
madbomber - A Kaboom! clone
tcs - Character set translator.
atari800 - Atari emulator for svgalib/X/curses
stella - Atari 2600 Emulator for X windows
xmess-x - X binaries for Multi-Emulator Super System
```

Wir finden verschiedene Pakete mit kurzen Beschreibungen. Um weitere Informationen über ein bestimmtes Paket zu erhalten, können wir folgendes machen:

```
root@linux / # apt-cache show stella

Package: stella
Priority: extra
Section: non-free/otherosfs
Installed-Size: 830
Maintainer: Tom Lear <tom@trap.mtview.ca.us>
Architecture: i386
Version: 1.1-2
Depends: libc6 (>= 2.1), libstdc++2.10, xlib6g (>= 3.3.5-1)
Filename: dists/potato/non-free/binary-i386/otherosfs/stella_1.1-2.deb
Size: 483430
MD5sum: 11b3e86a41a60fa1c4b334dd96c1d4b5
Description: Atari 2600 Emulator for X windows
 Stella is a portable emulator of the old Atari 2600 video-game console
 written in C++.  You can play most Atari 2600 games with it.  The latest
 news, code and binaries for Stella can be found at:
 http://www4.ncsu.edu/~bwmott/2600
```

Mit dieser Ausgabe erhalten Sie eine Menge Details über das Paket, das Sie installieren wollen (oder nicht wollen) inklusive der vollständigen Beschreibung des Pakets. Wenn das Paket schon installiert ist und es eine neuere Version gibt, bekommen Sie Informationen über beide Versionen. Beispiel:

```
root@linux / # apt-cache show lilo

Package: lilo
Priority: important
Section: base
Installed-Size: 271
Maintainer: Russell Coker <russell@coker.com.au>
Architecture: i386
Version: 1:21.7-3
Depends: libc6 (>= 2.2.1-2), debconf (>=0.2.26), logrotate
Suggests: lilo-doc
Conflicts: manpages (>>1.29-3)
Filename: pool/main/l/lilo/lilo_21.7-3_i386.deb
Size: 143052
MD5sum: 63fe29b5317fe34ed8ec3ae955f8270e
Description: LInux LOader - The Classic OS loader can load Linux and
others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text
screen)
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.

Package: lilo
Status: install ok installed
Priority: important
Section: base
Installed-Size: 190
Maintainer: Vincent Renardias <vincent@debian.org>
Version: 1:21.4.3-2
Depends: libc6 (>= 2.1.2)
Recommends: mbr
Suggests: lilo-doc
Description: LInux LOader - The Classic OS loader can load Linux and
others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text
screen)
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

Das erste in der Liste ist die neu verfügbare Version und das zweite die installierte Version. Für generellere Informationen über ein Paket können sie folgendes benutzen:

```
root@linux / # apt-cache showpkg penguin-command

Package: penguin-command
Versions:
1.4.5-1(...)(/var/lib/dpkg/status)
```

```
Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0)
libsdl1.1 (0 (null)) zlib1g (2 1:1.1.3)
Provides:
1.4.5-1 -
Reverse Provides:
```

Und um nur herauszufinden, von welchen Paketen es abhängt:

```
root@linux / # apt-cache depends penguin-command

penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

Zusammengefasst haben wir eine handvoll Waffen, die wir benutzen können, um den Namen des Paketes herauszufinden, das wir installieren wollen.

5.2 Paketnamen mit dpkg finden

Ein Weg, den Namen eines Pakets zu finden, ist, den Namen einer wichtigen Datei zu kennen, die sich in dem Paket befindet. Um zum Beispiel das Paket zu finden, welches eine bestimmte **.h** Datei enthält, die für das Kompilieren eines Programms benötigt wird, ist folgendes auszuführen:

```
root@linux / # dpkg -S stdio.h

libc6-dev: /usr/include/stdio.h
libc6-dev: /usr/include/bits/stdio.h
perl: /usr/lib/perl/5.6.0/CORE/nostdio.h
```

oder:

```
root@linux / # dpkg -S /usr/include/stdio.h

libc6-dev: /usr/include/stdio.h
```

Um den Namen installierter Pakete herauszufinden, was zum Beispiel zum Aufräumen der Festplatte nützlich sein kann, benutzen Sie:

```
root@linux / # dpkg -l | grep mozilla

ii  mozilla-browse 0.9.6-7        Mozilla Web Browser
```

Das Problem mit diesem Befehl ist, dass er Paketnamen **brechen** kann. Im obigen Beispiel ist der ganze Name des Pakets mozilla-browser. Um das Problem zu beheben, können Sie die Umgebungsvariable `COLUMNS` folgendermaßen benutzen:

```
root@linux / # COLUMNS=132 dpkg -l | grep mozilla

ii  mozilla-browser             0.9.6-7                      Mozilla Web
Browser - core and browser
```

oder die Beschreibung bzw. einen Teil dieser wie im folgenden:

```
root@linux / # apt-cache search "Mozilla Web Browser"

mozilla-browser - Mozilla Web Browser
```

5.3 Pakete nach Bedarf installieren

Sie kompilieren gerade ein Programm, und es gibt einen Fehler, da eine .h Datei gebraucht wird, die Sie nicht haben. Das Programm `auto-apt` kann Sie vor solchen Szenarios bewahren. Es fragt, ob es die benötigten Pakete installieren soll, nachdem es den betroffenden Prozess gestoppt hat und führt ihn fort, wenn die relevanten Pakete installiert sind.

Der Befehl sieht folgendermassen aus:

```
root@linux / # auto-apt run Kommando
```

Wobei **Kommando** das Kommando ist, das ausgeführt werden soll und evtl. nicht vorhandene Dateien benötigt. Beispiel:

```
root@linux / # auto-apt run ./configure
```

Es wird fragen, ob die benötigten Pakete installiert werden sollen, und `apt-get` automatisch aufrufen. Wenn X läuft, ersetzt eine grafische Oberfläche die übliche Text-Oberfläche.

`Auto-apt` funktioniert mit einer Datenbank welche aktuell gehalten werden muss, um effektiv zu funktionieren. Das erreicht man mit den Kommandos `auto-apt update`, `auto-apt updatedb` und `auto-apt update-local`.

5.4 Herausfinden, zu welchem Paket eine Datei gehört

Wenn ein Paket installiert werden soll und Sie nicht herausfinden können, wie es heißt, indem Sie mit `apt-cache` suchen, aber den Dateinamen des Programms oder einer Datei, die zu dem Paket gehört kennen, können Sie `apt-file` benutzen, um den Dateinamen zu finden. Das wird folgendermaßen gemacht:

```
root@linux / # apt-file search Dateinamen
```

Es funktioniert genau wie `dpkg -S`, es zeigt Ihnen aber auch nicht installierte Pakete, die die Datei enthalten.

Man kann es auch dazu benutzen, benötigte include-Dateien, die beim Kompilieren von Programmen fehlen, zu installieren, allerdings ist `auto-apt` eine wesentlich bessere Methode solche Fälle zu lösen, siehe Pakete nach Bedarf installieren.

Man kann auch den Inhalt von Paketen auflisten:

```
root@linux / # apt-file list Paketname
```

`apt-file` hat genau wie `auto-apt` eine Datenbank über die Dateien aller Pakete und diese muss aktuell gehalten werden:

```
root@linux / # apt-file update
```

Normalerweise benutzt `apt-file` die gleiche Datenbank wie `auto-apt`, sehen Sie Pakete nach Bedarf installieren.

5.5 Über Änderungen in Paketen informiert bleiben

Jedes Paket installiert in sein Dokumentationsverzeichnis (/usr/share/doc/Paketname) eine Datei mit Namen changelog.Debian.gz, welche die Liste der Änderungen gegenüber der letzten Version enthält. Sie können diese Dateien z. B. mit Hilfe von `zless` lesen, aber es ist nicht wirklich leicht, nach einem System-Upgrade nach dem changelog jedes aktualisierten Paketes zu suchen.

Es gibt aber eine Möglichkeit, diese Aufgabe zu automatisieren mit Hilfe eines Werkzeugs mit Namen `apt-listchanges`. Hierfür muss das Paket `apt-listchanges` erst einmal installiert werden. Während der Installation übernimmt `Debconf` die Installation. Beantworten Sie die Fragen nach Ihren Bedürfnissen.

Die Option **Soll apt-listchanges nach dem Anzeigen der Changelogs um eine Bestätigung bitten?** ist sehr nützlich, da es eine Liste der Änderungen jedes Paketes, das während eines Upgrades installiert wird, anzeigt und Ihnen die Möglichkeit bietet, diese vor dem Fortfahren einzusehen. Wenn Sie hier sagen, dass Sie nicht fortfahren möchten, gibt apt-listchanges einen Fehlercode zurück und `APT` bricht die Installation ab.

Nachdem `apt-listchanges` installiert wurde, zeigt es die Liste der Änderungen installierter Pakete an, wenn Pakete aus dem Netz (oder von einer CD oder gemounteten Partition) heruntergeladen werden, bevor sie installiert werden.

6 Das Arbeiten mit Quellpaketen

6.1 Herunterladen von Quellpaketen

In der Welt der freien Software ist es üblich, den Quellcode zu studieren oder auch Korrekturen an fehlerhaftem Code vorzunehmen. Um dieses zu tun, muss der Quellcode des Programms heruntergeladen werden. Das APT-System bietet eine einfache Möglichkeit, den Quellcode der vielen Programme der Distribution einschließlich aller für das Erstellen eines .deb des Programms nötigen Dateien zu beziehen.

Eine andere übliche Anwendung für Debian-Quellen ist es eine aktuellere Version eines Programms aus der Distribution **unstable** zum Beispiel in **stable** zu benutzen. Das Paket **gegen stable** zu kompilieren erzeugt ein Paket mit Abhängigkeiten, die auf die Pakete aus **stable** ausgerichtet sind.

Hierfür sollte der `deb-src`-Eintrag in Ihrer /etc/apt/sources.list auf **unstable** zeigen. Er sollte ausserdem aktiviert sein, d.h. eventuelle Kommentarzeichen vor der Zeile müssen entfernt werden (siehe Abschnitt Die Datei /etc/apt/sources.list).

Um ein Quellpaket herunterzuladen, benutzen Sie folgendes Kommando:

```
root@linux / # apt-get source Paketname
```

Drei Dateien werden daraufhin heruntergeladen: ein .orig.tar.gz, ein .dsc und ein .diff.gz. Im Falle von Paketen, die speziell für Debian erzeugt wurden, fällt das letzte weg und das erste hat kein orig im Namen.

Die Datei .dsc wird von `dpkg-source` benutzt, um das Quellpaket in das Verzeichnis Paketname-Version zu entpacken. In jedem heruntergeladenen Quellpaket befindet sich ein Verzeichnis debian/, welches die für das Bauen des .deb-Paketes nötigen Dateien enthält.

Um das Paket beim Herunterladen automatisch zu erzeugen, fügen Sie einfach `-b` zur Kommandozeile hinzu:

```
user@linux / $ apt-get -b source Paketname
```

Wenn Sie sich dazu entscheiden, das Paket noch nicht beim Herunterladen zu erzeugen, können Sie dieses später nachholen mittels

```
user@linux / $ dpkg-buildpackage -rfakeroot -uc -b
```

in dem Verzeichnis, welches für das Paket nach dem Herunterladen erstellt wurde. Um das zuvor erzeugte Paket zu installieren, muss man den Paketmanager direkt einsetzen:

```
root@linux / # dpkg -i Datei.deb
```

Es besteht ein Unterschied zwischen der Methode `apt-get source` und den anderen Methoden von `apt-get`. Die `source`-Methode kann von normalen Benutzern **ohne Root-Rechte** benutzt werden. Die Dateien werden in das Verzeichnis heruntergeladen, aus dem das `apt-get source Paket` aufgerufen wurde.

6.2 Für das Kompilieren eines Quellpaketes nötige Pakete

Normalerweise müssen sich spezielle Bibliotheken auf dem System befinden, um ein Quellpaket zu kompilieren. Alle Quellpakete haben ein Feld mit Namen **Build-Depends** in ihrer Kontrolldatei, welches die Namen der zusätzlichen Pakete enthält, die für das Erzeugen des Paketes aus dem Quellcode nötig sind.

`APT` bietet eine einfache Möglichkeit diese Pakete herunterzuladen. Führen Sie einfach `apt-get build-dep Paket` aus, wobei **Paket** für den Namen des Pakets, welches Sie erzeugen wollen steht. Beispiel:

```
root@linux / # apt-get build-dep gmc

Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  comerr-dev e2fslibs-dev gdk-imlib-dev imlib-progs libgnome-dev
libgnorba-dev
  libgpmg1-dev
0 packages upgraded, 7 newly installed, 0 to remove and 1  not upgraded.
Need to get 1069kB of archives. After unpacking 3514kB will be used.
Do you want to continue? [Y/n]
```

Die Pakete, die hier installiert werden, werden gebraucht, um gmc korrekt zu erzeugen. Beachten Sie jedoch, dass dieses Kommando sich nicht um das Quellpaket selbst kümmert, welches Sie bauen möchten. Sie müssen hierfür zusätzlich `apt-get source` ausführen.

Falls Sie nur feststellen möchten, welche Pakete zum Bau eines bestimmten Paketes benötigt werden, ist eine Variante des Kommandos `apt-cache show` (siehe Informationen über Pakete), die neben anderer Information die Zeile Build-Depends aufführt, die ihrerseits die erforderlichen Pakete auflistet.

```
root@linux / # apt-cache showsrc Paket
```

7 Der Umgang mit Fehlern

7.1 Häufige Fehler

Fehler wird es immer geben. Viele werden durch unachtsame Benutzer verursacht. Im folgenden finden Sie eine Liste mit häufig gemeldeten Fehlern und wie Sie mit ihnen umgehen sollten.

Wenn Sie eine Nachricht erhalten, die aussieht wie die im unteren Beispiel, bei dem Versuch `apt-get install Paket` auszuführen ...

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/
Packages'
(/var/state/apt/lists/people.debian.org_%7ekov_debian_unstable_Packages)
- stat (2 No such file or directory)
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

haben Sie vergessen apt-get update nach Ihrer letzten Änderung in der /etc/apt/sources.list auszuführen.

Wenn folgender Fehler auftritt...

```
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission
denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you
root?
```

nach dem Versuch, irgend eine andere `apt-get`-Methode auszuführen als source, haben Sie keine Root-Rechte, d.h. Sie haben sie als normaler Benutzer ausgeführt.

Der gleiche Fehler wie oben tritt auf, wenn versucht wird, zweimal `apt-get` gleichzeitig auszuführen oder auch, wenn versucht wird `apt-get` auszuführen während ein `dpkg`-Prozess läuft. Die einzige Methode, die simultan zu anderen ausgeführt werden darf, ist die `source`-Methode.

Wenn eine Installation mitten im Prozess abbricht und Sie merken, dass es nicht länger möglich ist, Pakete zu installieren oder zu entfernen, versuchen Sie diese zwei Befehle auszuführen:

```
root@linux / # apt-get -f install dpkg --configure -a
```

Danach versuchen Sie es erneut. Es kann nötig sein, den zweiten der beiden Befehle mehr als einmal auszuführen. Dies ist eine wichtige Information für die Abenteurer, die **unstable** benutzen.

Tritt **E: Dynamic MMap ran out of room** beim Ausführen von `apt-get update` auf, sollte die folgende Zeile der /etc/apt/apt.conf hinzugefügt werden:

/etc/apt/apt.conf

```
APT::Cache-Limit 10000000;
```

7.2 Wo gibt es Hilfe?

Wenn Sie sich von Zweifeln geplagt fühlen, ziehen Sie die umfangreiche Dokumentation des Debian-Paketsystems zu Rate. `--help` und Manpages können eine enorme Hilfe sein, genau wie die Dokumentation in den Verzeichnissen in /usr/share/doc ebenso wie in /usr/share/doc/apt.

Wenn diese Dokumentation nicht ausreicht, um Ihre Probleme zu beseitigen, versuchen Sie es auf den Debian-Mailinglisten. Mehr Informationen über die speziellen Benutzer-Listen gibt es auf der Debian-Webseite: http://www.debian.org.

Natürlich sind diese Listen und Hilfen nur für Debian-Benutzer; Benutzer anderer Systeme werden von der Gemeinschaft ihrer Distribution bessere Hilfe erlangen.

8 Welche Distributionen unterstützen APT?

Hier finden Sie die Namen einiger Distributionen, die APT unterstützen:

* Debian GNU/Linux (http://www.debian.org) - Für diese Distribution wurde APT entwickelt
* Debian-Derivate wie (K)Ubuntu (http://www.ubuntu.com, http://www.kubuntu.com)
* Conectiva (http://www.conectiva.com.br) - Die erste Distribution, die APT mit RPM benutzt
* Mandrake (http://www.mandrake.com)
* PLD (http://www.pld.org.pl)
* Vine (http://www.vinelinux.org)
* APT4RPM (http://apt4rpm.sf.net)
* Alt Linux (http://www.altlinux.ru/)
* Red Hat (http://www.redhat.com/)
* Sun Solaris (http://www.sun.com/)
* SuSE (http://www.suse.de/)
* Yellow Dog Linux (http://www.yellowdoglinux.com/)
* Yoper (http://www.yoper.de/, http://www.yoper.com/)