

Paketverwaltung von DEB Paketen

Johannes Knopp

29.04.2008

Übersicht

- 1 Pakete allgemein
- 2 Aufbau eines .deb Pakets I
- 3 Arbeiten mit Paketen I
 - dpkg
 - (De)Installations- und Updatevorgänge
- 4 Aufbau eines .deb Pakets II
 - Abhängigkeiten
 - Paketprioritäten
- 5 Arbeiten mit Paketen II
 - Bezugsquellen von Paketen
 - Probleme bei Abhängigkeiten
 - apt - Advanced packaging tool
 - Grafische Paketverwaltung mit Synaptic
- 6 Zusammenfassung

Pakete allgemein

- Auf jedem Betriebssystem laufen verschiedene Programme
- Benutzer wollen neue Software installieren
- Installierte Programme sollen auf dem neuesten Stand sein
- Linux bietet Funktionen mithilfe von Paketen

Pakete – Wozu?

Idee der Pakete folgt Linux grundlegendem Konzept des *Setzkastenprinzips*:

Setzkastenprinzip

Probleme werden in Teilprobleme heruntergebrochen und auf dieser Ebene gelöst. Daraus ergeben sich Programmbausteine (Tools/Kommandos), die wieder in größeren Teilen Verwendung finden können.

Diese Herangehensweise bietet zahlreiche Vorteile!

Pakete – Wozu?

- Robustheit: Es ist möglichst schwer, das System ungewollt zu korrumpieren

¹Zitat und Auflistung von <http://www.schlittermann.de/deb-intern/dpkg/>

Pakete – Wozu?

- Robustheit: Es ist möglichst schwer, das System ungewollt zu korrumpieren
- Einfache Bedienung: Keine Hürden für Anfänger – Fortgeschrittene haben dennoch komplette Funktionalität

¹Zitat und Auflistung von <http://www.schlittermann.de/deb-intern/dpkg/>

Pakete – Wozu?

- Robustheit: Es ist möglichst schwer, das System ungewollt zu korrumpieren
- Einfache Bedienung: Keine Hürden für Anfänger – Fortgeschrittene haben dennoch komplette Funktionalität
- Upgrade-Fähigkeit: Linux dynamische Entwicklung bringt häufig neue Programmversionen hervor. Diese können problemlos integriert werden und erfordern kein neues Booten

¹Zitat und Auflistung von <http://www.schlittermann.de/deb-intern/dpkg/>

Pakete – Wozu?

- Robustheit: Es ist möglichst schwer, das System ungewollt zu korrumpieren
- Einfache Bedienung: Keine Hürden für Anfänger – Fortgeschrittene haben dennoch komplette Funktionalität
- Upgrade-Fähigkeit: Linux dynamische Entwicklung bringt häufig neue Programmversionen hervor. Diese können problemlos integriert werden und erfordern kein neues Booten
- Verwaltung des gesamten Systems: "Das Paket-System soll in der Lage sein, das Debian-System vollständig zu verwalten. Insbesondere soll kein extra Basis-System erstellt werden, daß sich der Paketverwaltung entzieht und dessen Inhalt deshalb nicht einfach upgradebar ist." ¹

¹Zitat und Auflistung von <http://www.schlittermann.de/deb-intern/dpkg/>

Pakete - .deb Format

- Am weitesten verbreitete Paketformate: *.rpm* und *.deb*
- Dieser Vortrag behandelt *.debs*
- Sind Grundlage aller Debian basierten Distributionen: Ubuntu, MEPIS, Dreamlinux, Damn Small Linux, Xandros, Knoppix, Linspire, sidux, Kanotix. . .
- Stehen in der Regel unter einer freien Lizenz wie GPL, LGPL, BSD, oder Artistic

Aufbau eines .deb Paketes I

- DEB-Pakete sind mit **ar** gepackt
- Dateiname:
`<foo>_<VersionNumber>-<DebianRevisionNumber>_<DebianArchitecture>.deb`
wobei *foo* den Paketnamen darstellt.
- Entpackt ergeben sich drei Dateien

Aufbau eines .deb Paketes I

- 1 **debian-binary**: Die Versionsnummer des verwendeten Paketformats (momentan 2.0)
- 2 **control.tar.gz**: Archiv mit Dateien zur Installation und Informationen zu Abhängigkeiten
- 3 **data.tar.gz**: enthält eigentliche Programmdateien

Arbeiten mit Paketen I - dpkg

- Debian package management system - dpkg
- Low Level Werkzeug
- Fähigkeiten:
 - Pakete installieren
 - Pakete entfernen
 - Pakete verwalten
 - (Pakete bauen)

Arbeiten mit Paketen I - dpkg

Installieren

```
dpkg --install <paketname>
```

Deinstallieren

```
dpkg --remove <paketname>
```

remove lässt Konfigurationsdateien auf dem System. Zur kompletten Entfernung *--purge* übergeben!

Installations- und Updatevorgang

6 Phasen bei Installation/Update:

- 1 *Prerm*: Vor dem Entfernen vom System. Das alte Paket räumt Logdateien auf oder beendet bestimmte Dienste
- 2 Backups der Originale werden angelegt. Dateien des neuen Paketes werden ausgepackt und überschreiben alte Originale.
- 3 *Preinst*: Vor dem Entpacken auf die Festplatte. Letzte Chance für neues Paket, Dinge zu prüfen, die durch normalen Abhängigkeitsbeziehungen nicht erschlagen werden können.
- 4 *Postrm*: Letzte Aufräumarbeiten des dahinscheidenden Pakets.
- 5 Backups alter Originale werden entfernt: Point of No Return
- 6 *Postinst*: Nach dem Entpacken. Letzte Installationsarbeiten, z.B. endgültige Integration in das SysV Init-System, Abfeuern von Daemons. . .

Deinstallationsvorgang

Deinstallieren ist etwas einfacher:

- 1 *Prerm* wird aufgerufen.
- 2 Alle zum Paket gehörenden Files werden entfernt, Konfigurationsdateien bleiben erhalten.
- 3 *Postrm* wird ausgeführt.
- 4 Konfigurationsdateien werden entfernt (bei *purging*)

Da Konfigurationsdateien oft aufwändig per Hand erstellt wurden, spielen sie eine Sonderrolle

Paketzustand

Pakete können installiert sein oder nicht. Alle Zustände:

- *Not installed* **n**: nicht installiert
- *Installed* **i**: installiert
- *Config-Files* **c**: nicht mehr installiert, aber die Config-Dateien sind noch vorhanden
- *Unpacked* **u**: ausgepackt, aber noch nicht konfiguriert
- *Failed-Config* **f**: Fehler bei Konfiguration aufgetreten
- *Half-installed* **h**: Paket unvollständig installiert

Aufbau eines .deb Pakets II

Pakete und dessen einfache Funktionen sind nun klar, betrachten wir nun die Details. Es war die Rede davon, dass das Archiv **control.tar.gz** in einem .deb Paket Abhängigkeiten auflistet und der Installation dient. Was hat es mit Abhängigkeiten auf sich?

Abhängigkeiten I

- **Pre-Depends:** Aufgeführte Pakete müssen fertig installiert sein. Die meisten Pakete setzen z.B. *libc* voraus
- **Depends:** Paket kann nur installiert werden, wenn aufgeführte Pakete auch installiert sind bzw. werden. Die Reihenfolge der Installation ist dabei egal
- **Recommends:** Paket empfiehlt sehr stark, die aufgeführten Pakete zu installieren. Betrifft Pakete die in einer üblichen Installation zusammengehören. *Sendmail* geht davon aus, daß normalerweise auch ein "mail-reader" notwendig ist.
- **Suggests:** Paket schlägt die Installation weiterer Pakete vor. Der Vorschlag ist nicht zwingend, sollte aber für gewünschte Funktionalität beachtet werden. Ein Beispiel: Der Mailreader *mutt* schlägt vor, auch *pgp* zu installieren. Funktionieren würde er auch ganz gut ohne. . .

Abhängigkeiten II

- **Conflicts:** Die Installation wird abgelehnt, falls eines der hier aufgeführten Pakete installiert ist. Auf einem System darf es z. B. nur **ein** Print-System geben.
- **Replaces:** Das Paket ersetzt ein anderes. Zum Beispiel ersetzt die Release-Version eine zuvor installierte Betaversion eines Programms.
- **Provides:** Paketauflistung dessen Funktionalität das jeweilige Paket auch zur Verfügung stellt. Z.B. gibt es mehrere Pakete, die hier einen mail-reader aufführen (*elm*, *mutt*, *mh*, *emacs*, ...). Damit ist es für andere Pakete möglich, sich vom Vorhandensein **irgendeines** Mail-Readers abhängig zu machen. Mehr dazu findet man unter dem Stichwort *virtuelle Pakete/virtual packages*.

Paketprioritäten

Stehen auch in der *control*-Datei und beschreiben Wichtigkeit des Pakets für ein Linux-System

- **Required**-Pakete sind nötig, damit System ordentlich läuft (z.B. Programme zum Reparieren des Systems).
- **Important**-Pakete sollten auf jedem Unix-like System gefunden werden. Keine großen Anwendungen wie *X11*, eher für Infrastruktur.
- **Standard**: Standardpakete eines Linux-Systems. Für den Anwender grundlegende Pakete sind so markiert (*OpenSSH, exim, mutt ...*)
- **Optional**: Weitere Pakete für den Anwender. Unter anderem *X11* oder eine *TEX* Distribution.
- Mit **Extra** sind Pakete gekennzeichnet, die z.B. im Konflikt mit Paketen höherer Priorität stehen oder spezielle Anforderungen haben. Sollten nur installiert werden, wenn der Anwender auch weiß, was es damit auf sich hat.

Arbeiten mit Paketen II

Mit dpkg können wir nun Pakete installieren und entfernen. Zwei Fragen:

- 1 Woher kann ich Pakete bekommen?
- 2 Was mache ich, wenn die Abhängigkeiten von einem Paket nicht erfüllt sind?

Bezugsquellen von Paketen

Quellen für Pakete sind in `/etc/apt/sources.list` aufgelistet

Probleme bei Abhängigkeiten

- Stößt dpkg beim Installieren auf ungelöste Abhängigkeiten bricht es ab

Probleme bei Abhängigkeiten

- Stößt dpkg beim Installieren auf ungelöste Abhängigkeiten bricht es ab
- Benutzer muss fehlende Pakete herunterladen und installieren

Probleme bei Abhängigkeiten

- Stößt dpkg beim Installieren auf ungelöste Abhängigkeiten bricht es ab
- Benutzer muss fehlende Pakete herunterladen und installieren
- Diese haben evtl. wieder Abhängigkeiten, die wieder Abhängigkeiten haben, die wieder. . . ⇒ *Dependency Hell*

Probleme bei Abhängigkeiten

- Stößt dpkg beim Installieren auf ungelöste Abhängigkeiten bricht es ab
- Benutzer muss fehlende Pakete herunterladen und installieren
- Diese haben evtl. wieder Abhängigkeiten, die wieder Abhängigkeiten haben, die wieder... ⇒ *Dependency Hell*
- Lösung: Ein Programm zur Paketverwaltung: **apt**

apt - Advanced packaging tool

- Kann Pakete installieren und entfernen
- Erkennt Abhängigkeiten und löst sie automatisch auf

apt - Advanced packaging tool

- *apt-get update* aktualisiert die Liste der verfügbaren Pakete.
- *apt-get install <paketname>*
- *apt-get remove/purge <paketname> --purge* werden alle entsprechenden Konfigurationsdateien mit gelöscht.
- *apt-get -u upgrade* bringt die installierten Programme auf den neuesten Stand.
- *apt-get -u dist-upgrade* bringt das System auf den Stand eines neuen größeren Releases.
- Suche mit *apt-cache search <begriff>*
- Paketdetails anzeigen mit *apt-cache show <paketname>*

Arbeiten mit Paketen II - apt

```
ponk@foster:~$ aptitude install scummvm
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be installed:
  ladcca2{a} libfluidsynth1{a} libjack0.100.0-0{a} scummvm
0 packages upgraded, 4 newly installed, 0 to remove and 14 not upgraded
Need to get 2530kB of archives. After unpacking 7770kB will be used.
Do you want to continue? [Y/n/?]
```

Arbeiten mit Paketen II - apt

- Deinstallation: apt ignoriert Pakete, die aufgrund von Abhängigkeiten installiert wurden. ⇒ Verwaiste Pakete
- Lösung: **aptitude**

Arbeiten mit Paketen II - aptitude

- Aptitude ist ein Front-End zu apt mit zusätzlichen Fähigkeiten
- Einfache GUI
- Merkt sich ausgeführte Schritte
- Deinstalliert Pakete, die nur wegen Abhängigkeiten auf dem System sind, sobald das von ihnen abhängige Paket deinstalliert wird
- Beachtet empfohlene Pakete

Grafische Paketverwaltung mit Synaptic

- Front-End zu apt (nicht aptitude!)
- Einfacherer Einstieg für Neulinge
- Gut für weniger konsolenorientierte Benutzer

Zusammenfassung

- Paketverwaltung ermöglicht sehr dynamische Systemverwaltung
- Nur benötigte Pakete werden installiert, kein (bzw. kaum) Overhead
- Abhängigkeiten müssen gelöst werden, dafür gibt es verschiedene Möglichkeiten (Per Hand, mit apt, mit aptitude..)

Quellen

http://www.debian.org/doc/FAQ/ch-pkg_basics

<http://www.debian.org/doc/manuals/debian-faq/ch-pkgtools.en.html>

<http://nixdoc.net/man-pages/Linux/dpkg.8.html>

<http://www.schlittermann.de/deb-intern/dpkg/>

<http://www.debian.org/doc/manuals/apt-howto/>

<http://www.youtube.com/watch?v=lFzPrzY2KFM&v3>

http://en.wikipedia.org/wiki/Dependency_hell

http://en.wikipedia.org/wiki/Aptitude_%28program%29

<http://en.wikipedia.org/wiki/Dpkg>

http://en.wikipedia.org/wiki/List_of_Linux_distributions#Debian-based

<http://de.wikipedia.org/wiki/Debian-Bin%C3%A4rpaket>

<http://www.newlinuxuser.com/howto-use-dpkg-to-install-deb-files/>

<http://pthree.org/2007/08/12/aptitude-vs-apt-get/>