3.2 Grundkonfiguration/Paketverwaltung

3.2.1 Wozu Software- und Paket-Verwaltung?

Ursprünglich funktioniert das Einspielen von neuer Software auf einem Rechnersystem in der Regel so: Man bekommt einen Datenträger sowie eine (oft sehr kurze) schriftliche Anleitung, was nach dem Einlegen des Datenträgers zu tun ist. Die Anleitung führt man aus, womit die Software installiert ist. Zum Deinstallieren benutzt man (falls möglich) dieselbe Anleitung. Oft befindet sich auf dem Datenträger ein Installationsprogramm, das dann die Programmdateien und die Konfigurationsdaten an die richtigen Stellen kopiert.

Diese Vorgehensweise ist einfach, hat aber Nachteile. So geschieht die Installation ganz ohne Aufsicht des Betriebssystems. Das Installationsprogramm macht seine Sache meist gut. Beim Deinstallieren werden häufig nicht alle Spuren der Installation beseitigt, so dass im Laufe der Zeit ein stark mutiertes System entsteht.

Bei einer zentralen Software- und Paketverwaltung ist das anders. Jede Installation verläuft für den Administrator gleich. Installation und Deinstallation verlaufen spiegelbildlich. Alles, was installiert wurde, wird auf Wunsch auch wieder entfernt. Die Installation läuft unter Aufsicht des Paket-Managers des Betriebssystems (Abbildung 1. Abhängigkeiten zwischen Paketen werden er-

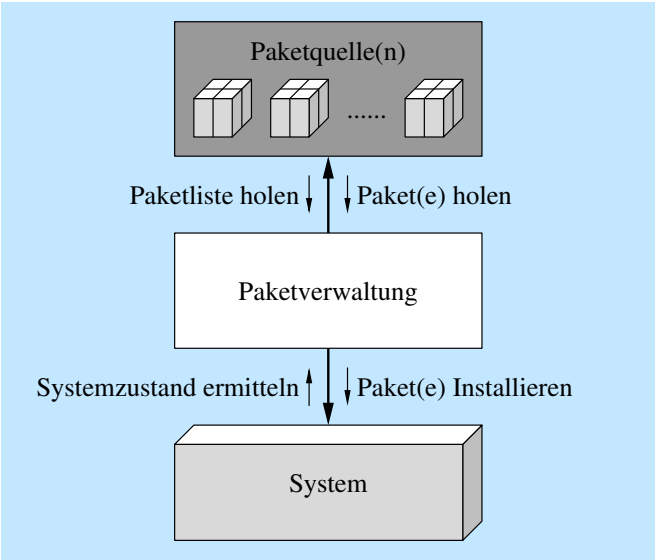

Abbildung 1: Paketverwaltung

kannt und, wenn möglich, aufgelöst. Auf Wunsch können alle Bestandteile des Systems aktualisiert werden. Dazu werden Quellen im Netz oder auf Massenspeichern bereitgestellt.

3.2.2 Was ist ein Paket?

Ein Paket enthält in der Regel ein Programm oder eine Programmsammlung, die eine bestimmte Aufgabe löst. Es umfasst damit eine Binärdatei, eine oder mehrere Konfigurationsdateien und meistens auch eine Manual-Page. Es ist aber sinnvoll, noch weitere Informationen in das Paket zu stecken, nämlich eine Beschreibung für menschliche Nutzer und Anweisungen für den Paket-Manager.

Ein oft benutztes Format für Pakete ist das deb-Format[1][2]. Benutzt wird es zum Beispiel bei den Distributionen Debian und Ubuntu. Pakete in diesem Format (erkennbar an der Datei-Endung .deb) liegen auf verschiedenen Servern weltweit. Hier ist eines davon:

[1] auch dpkg-Format genannt

[2] Ein anderes Format ist das rpm-Format, benutzt zum Beispiel bei Red Hat und SuSE. Und es gibt noch weitere.

Terminal
```
schueler@debian964:~$ wget http://ftp.de.debian.org/debian/pool/main\
/j/joe/joe_3.7-2.3_amd64.deb
...
```

Der Paketdateiname setzt sich wie folgt zusammen:

- `joe` ist der Paketname. Installierte Pakete werden unter ihrem Namen angesprochen.
- `3.7` ist die Version des Programmautors.
- `2.3` ist die Release des Paketbetreuers bei Debian.
- `amd64` ist die Architektur, für den das Paket compiliert wurde.
- `deb` ist die Kennung für ein Debian-Paket.

Auch ohne Paketmanager kann man in das Paket hineinsehen (Abbildung 2). Es wurde mit

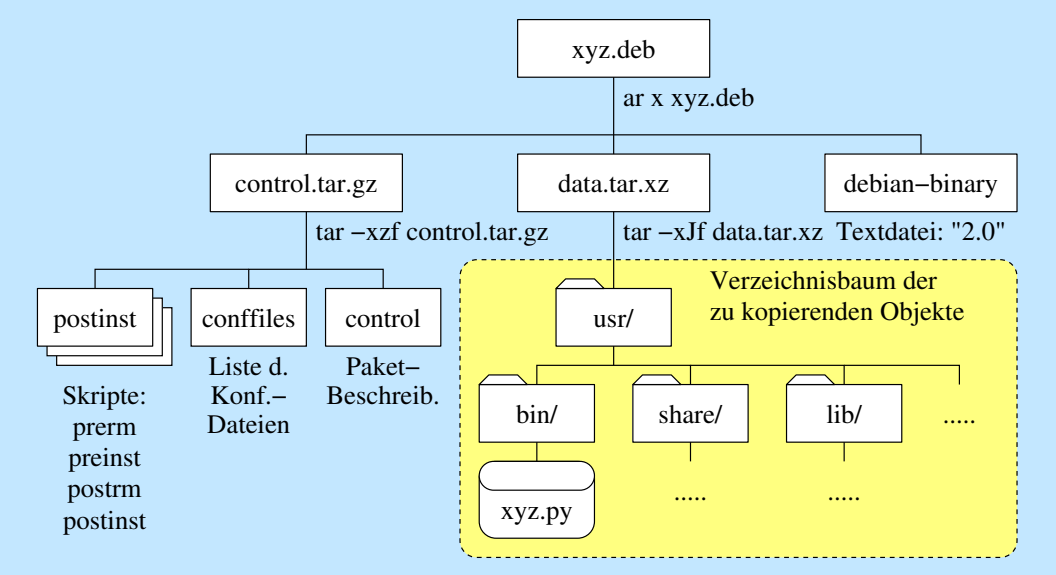

Abbildung 2: Innenaufbau eines deb-Paketes

dem Programm `ar` gepackt und kann damit auch wieder entpackt werden:

Terminal
```
schueler@debian964:~$ ar x joe_3.7-2.3_amd64.deb
schueler@debian964:~$ ls -l
...
-rw-r--r--  1 schueler schueler   1783 Apr 24 12:12 control.tar.gz
-rw-r--r--  1 schueler schueler 493183 Apr 24 12:12 data.tar.gz
-rw-r--r--  1 schueler schueler      4 Apr 24 12:12 debian-binary
```

In `debian-binary` findet man die Version des Paketformats (hier 2.0). In `data.tar.gz` liegen die Paketinhalte, also zum Beispiel die Binärdatei Konfiguration und Manual-Page. In `control.tar.gz` liegen die Anweisungen für den Paketmanager:

Terminal
```
schueler@debian964:~$ tar -xzvf control.tar.gz
./postrm
./postinst
```

```
./control
./prerm
./conffiles
./preinst
schueler@debian964:~$ cat control
Package: joe
Version: 3.7-2.3
Architecture: amd64
Maintainer: Josip Rodin <joy-packages@debian.org>
Installed-Size: 1305
Depends: libc6 (>= 2.7), libncurses5 (>= 5.5-5~), libtinfo5
Breaks: jupp (<< 3.1.18-2~)
Section: editors
Priority: optional
Homepage: http://joe-editor.sourceforge.net/
Description: user friendly full screen text editor
...
```

In der Datei `control` liegt, wie man sieht, die Paketbeschreibung. Jede Zeile hat einen Aufbau wie eine HTTP-Headerzeile. Interessant sind die beschriebenen Abhängigkeiten: Das Paket braucht das Paket libc6 mindestens in Version 2.7, dazu libncurses und libtinfo5. Es verträgt sich nicht mit frühen Versionen des Pakets jupp. Bei den Abhängigkeiten gibt es Stufen:

a) Depends=nötig: Ohne dieses Paket läuft das gewünschte Paket nicht

b) Recommends=empfohlen: Ohne dieses Paket läuft das gewünschte Paket nicht wie erwartet

c) Suggests=vorgeschlagen: Es ist sinnvoll, dieses Paket auch zu installieren

d) Extends=erweitert: Mit diesem Paket kann man aus dem gewünschten Paket noch mehr herausholen

In `conffiles` ist eine Liste der Konfigurationsdateien gegeben. Sie müssen bei Updates vorsichtig behandelt werden, und beim Löschen eines Paketes kann es sinnvoll sein, sie zu erhalten; denn manchmal steckt in Konfigurationsdateien viel Arbeit.

Terminal

```
schueler@debian964:~$ tar -xzvf data.tar.gz
...
./usr/share/man/man1/rjoe.1.gz
./usr/share/man/man1/jpico.1.gz
./usr/bin/jpico
./usr/bin/rjoe
```

Die Dateien in `data.tar.gz` werden bei der Installation ganz stumpf in das entsprechende Verzeichnis kopiert. Beim Löschen werden sie entsprechend gelöscht (Ausnahme: Konfigurationsdateien).

Die Dateien `postrm`, `postinst`, `prerm` und `preinst` sind Skripte, die vor bzw. nach diesem Kopier- oder Löschvorgang ausgeführt werden.

3.2.3 Paketverwaltung mit dpkg

Mit dem Paketmanager `dpkg` kann man nun solche Pakete verwalten, ohne mit `ar`, `tar` und `gzip` umgehen zu müssen (Abbildung 3).

Zunächst kann `dpkg` mit lokal vorhandenen Paketdateien umgehen:

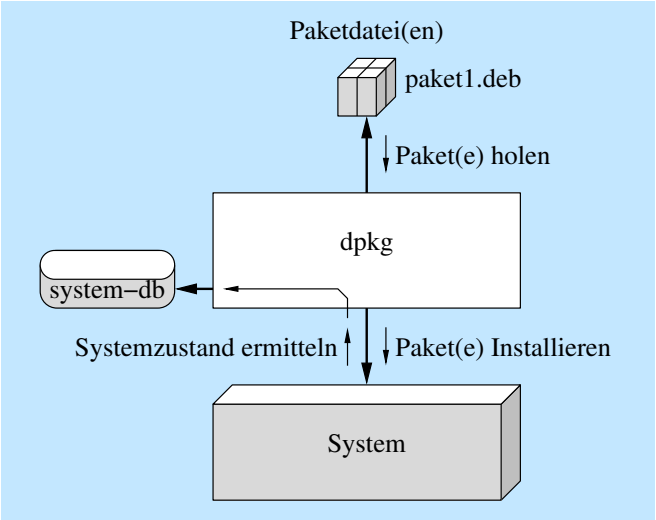

Abbildung 3: Paketverwaltung mit dpkg

Terminal
```
root@debian964:~# dpkg -i joe_3.7-2.3_amd64.deb # --install
```

Stattdessen kann man auch zuerst die Option `--unpack` und dann die Option `--configure` aufrufen.

Wo bekommt man solche deb-Dateien her? Man kann eine Debian-DVD einbinden und dann mit `find` suchen:

Terminal
```
schueler@debian964:~$ find /media/dvd -name "joe*.deb"
```

`dpkg` kann aber auch auf die bereits installierten Pakete auf dem System zugreifen. Dazu ruft es den Befehl `dpkg-query` mit denselben Optionen auf:

Terminal
```
schueler@debian964:~$ dpkg -l "*foo*" | cut -b1-80
    Name           Version      Architektur  Beschreibung
+++-==============-============-============-=========================
un  foo2zjs        <keine>      <keine>      (keine Beschreibung vorha
ii  foomatic-db-co 20161201-1   all          OpenPrinting printer supp
rc  foomatic-filte 4.0.17-5+deb i386         OpenPrinting printer supp
```

Die ersten beiden Spalten geben für jedes Paket den Status an, und zwar Wunsch und Wirklichkeit. Tabelle 1 gibt einen Überblick. Das Paket `foo2zjs` ist also nicht installiert (n), und es ist auch

Zielzustand		Aktueller Zustand	
u	unknown	n	not installed
i	install	i	installed
h	hold	c	config-files only
r	remove	u	unpack only
p	purge		

Tabelle 1: Paketzustände bei dpkg

keine Vormerkung bekannt (u). Dagegen ist das Paket `foomatic-db-co` installiert, und das soll es bleiben. Das Paket `foomatic-filte` soll gelöscht sein (r), und es sind tatsächlich nur noch die Konfigurationsdateien übrig (c). Es gibt offenbar zwei Arten des Löschens:

a) `remove` löscht das Paket, behält aber die Konfigurationsdateien

b) `purge` löscht das Paket mitsamt aller Konfigurationsdateien

Das Programm `dpkg` hat (als Front-End zu `dpkg-query`) noch einige weitere Möglichkeiten:

Terminal
```
schueler@debian964:~$ dpkg -L joe # --listfile,
                                  # liste alle Dateien d.Paketes auf
schueler@debian964:~$ dpkg -s joe # --status,
                                  # welchen Status hat das Paket?
schueler@debian964:~$ dpkg -S /etc/joe#--search,
                                  # zu welchem Paket gehört d.Datei?
```

`dpkg` (bzw. `dpkg-query`) liest dazu die sogenannte System-Datenbank, diese ist gespeichert in `/var/lib/dpkg/status` und in `/var/lib/dpkg/info/`.

Auch zum Packen und Entpacken von Paketdateien eignet sich `dpkg`. Für den Fall arbeitet es als Front-End für das Programm `dpkg-deb`:

Terminal
```
schueler@debian964:~$ dpkg -c joe_3.7-2.3_i386.deb # --contents,
                                 # Inhalte anzeigen
schueler@debian964:~$ dpkg -x joe_3.7-2.3_i386.deb /tmp # --extract,
                                 # entpacken nach /tmp
schueler@debian964:~$ dpkg -f joe_3.7-2.3_i386.deb Section # --field,
                                 # Feld Section anz.
schueler@debian964:~$ dpkg -I joe_3.7-2.3_i386.deb # --info,
                                 # komplette Information anz.
```

Die oben genannten Abhängigkeiten zwischen Paketen erkennt `dpkg` von selbst. Abhängigkeitsprobleme können bei Installation, beim Löschen und manchmal sogar beim Aktualisieren von Paketen auftreten:

Terminal
```
root@debian964:~# dpkg --remove netbase
dpkg: Abhängigkeitsprobleme verhindern Entfernen von netbase:
 libwww-perl hängt ab von netbase.
 exim4-base hängt ab von netbase.
 ...
```

Hier muss man also vorher `libwww-perl` und `exim4-base` löschen, damit man `netbase` löschen kann. In diesem Fall verhindert `dpkg` also, dass das System unbenutzbar wird (deshalb wird das Beispiel auch nicht weitergeführt). Bei `dpkg` muss das Löschen von Hand passieren, weil `dpkg` die Abhängigkeitsprobleme zwar selbst erkennt, aber nicht selbst automatisch auflöst.

Ursprünglich hat `dpkg` auch die Möglichkeit, mit einer Datenbank *verfügbarer* Pakete umzugehen, um so Updates und Installationen aus dem Netz zu ermöglichen. Dazu gibt es ein spezielles Programm, mit dem `dpkg` zusammenarbeitet, nämlich `dselect`, und eine dazugehörige Datenbank, nämlich `/var/lib/dpkg/available`. Das Programm `dselect` ist jedoch zugunsten von `APT` (s. u.) in den Hintergrund getreten.

3.2.4 Paketverwaltung mit APT, Konfiguration

Das APT (*advanced package tool*) ist eine Ergänzung zu `dpkg` um zwei Aspekte (Abbildung 4):

a) Download von Paketlisten und Paketen von diversen Quellen (Netz, lokaler Datenträger, Spiegelserver)

b) Selbständiges automatisches Auflösen von Abhängigkeitsproblemen

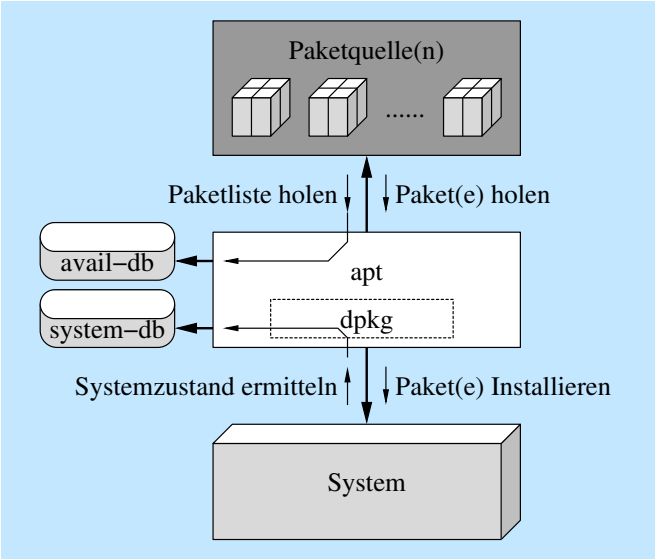

Abbildung 4: Paketverwaltung mit APT

Das APT stellt als Frontend das Programm `apt` für die wichtigsten Anwendungsfälle zur Verfügung. Für spezielle Situationen gibt es darüberhinaus die Programm `apt-get`, `apt-cdrom` und `apt-cache`.

Konfiguriert wird APT durch die Dateien in `/etc/apt`, vor allem `apt.conf` und `sources.list`. Zunächst sind die Netzwerk-Einstellungen in `apt.conf` wichtig. Wenn man das Debian-System hinter einem Proxy-Server betreibt, muss man das hier eintragen:

```
1 Acquire::http::proxy "http://vorname.name:geheim@10.1.1.3:8080/";
2 Acquire::ftp::proxy "http://vorname.name:geheim@10.1.1.3:8080/";
```

Eigentümer sollte `root` sein, die Berechtigungen `0600`. Bei der Datei `sources.list` gibt man ein, *woher die Paketlisten* eingelesen werden:

```
1  # Haupt-Paketquelle:
2  deb http://ftp.halifax.rwth-aachen.de/debian/ stretch main contrib non-free
3  deb-src http://ftp.halifax.rwth-aachen.de/debian/ stretch main contrib non-free
4  # Sicherheits-Updates:
5  deb     http://security.debian.org/ stretch/updates main contrib non-free
6  deb-src http://security.debian.org/ stretch/updates main contrib non-free
7  # Updates:
8  deb     http://ftp.halifax.rwth-aachen.de/debian/ stretch-updates main contrib
9  deb-src http://ftp.halifax.rwth-aachen.de/debian/ stretch-updates main contrib
10 # Vorgeschlagene Updates:
11 deb     http://ftp.halifax.rwth-aachen.de/debian/ stretch-proposed-updates\
12         contrib non-free main
13 deb-src http://ftp.halifax.rwth-aachen.de/debian/ stretch-proposed-updates\
14         contrib non-free main
```

Jede Zeile ist eine Paketquelle; die Felder sind durch Leerzeichen oder Tabulatoren getrennt. Dabei bedeutet:

a) `deb`
Paketart ist ein normales Debian-Paket. `deb-src` bedeutet Quellpaket.

b) `http://ftp.halifax.rwth-aachen.de/debian/`
URL, von der die Liste geladen wird. Der erste Teil der URL (hier: `http://`) erlaubt mehrere Möglichkeiten:

1) http: vom Netz mit http
2) ftp: vom Netz mit ftp
3) file: von lokal erreichbarer Datei
4) copy: wie file, aber mit lokaler Kopie
5) cdrom: vom Datenträger der Distribution, diese Zeile sollte mit `apt-cdrom` eingelesen werden

c) `stretch`
ist der Verzeichnisname für die Debian-Version. Versionen können hier auf zwei Arten angegeben werden, entweder als Codename (stretch) oder als Zustandsname (stable).

Folgende Zustandsnamen sind üblich[3]:

1) stable: stabile Arbeitsversion, z. Zt. Version 11
2) testing: Vorläufer der nächsten stabilen Arbeitsversion
3) unstable: Version mit neuesten Paketen, keine Nummer
4) oldstable: vorige stabile Arbeitsversion, z. Zt. Version 10

Die derzeitigen Codenamen findet man in Tabelle 2. Ein Schrägstrich am Ende bedeutete,

Version	Codename	Zustandsname Feb. 2023
7	wheezy	—
8	jessie	—
9	stretch	oldoldstable
10	buster	oldstable
11	bullseye	stable
12	bookworm	testing
13	trixie	—
14	forky	—

Tabelle 2: Gängige Debianversionen und Codenamen

dass alle Pakete im gleichen Verzeichnis lägen (flache Hierarchie!).

d) `main`
Software benutzen aus Lizenzkategorie: freie Software

e) `contrib`
Software benutzen aus Lizenzkategorie: freie Software, aber von nicht-freier Software abhängig

f) `non-free`
Software benutzen aus Lizenzkategorie: nach Debian-Maßstäben nicht vollkommen freie Software

Falls ein Paket von mehreren Quellen installiert werden könnte, wird das Paket mit der neuesten Version genommen[4].

[3] `https://en.wikipedia.org/wiki/Debian_version_history`
[4] Bei gleichen Versionen gewinnt die erste betreffende Zeile.

3.2.5 Paketverwaltung mit APT, Programme

APT hat für die verfügbaren Pakete eine eigene Datenbank. Sie liegt im Verzeichnis `/var/cache/apt/` in den Binärdateien `pkgcache.bin` (normale Pakete) und `srcpkgcache.bin` (Quellpakete). Die Datenbank wird aktualisiert durch den Befehl:

Terminal
```
root@debian964:~# apt update
```

Jetzt können eventuelle Programm-Updates erkannt und neue Paketdateien heruntergeladen und installiert werden. Diese drei Aktionen passieren mit:

Terminal
```
root@debian964:~# apt upgrade
```

Die neuen Paketversionen wurden kopiert nach `/var/cache/apt/archives`. Dann wurden sie mit dem Aufruf `dpkg -i` installiert.

Sollten durch die Aktualisierung neue Abhängigkeiten hinzukommen, die nicht aufgelöst werden können (kommt praktisch nur dann vor, wenn man selbst umfassende Änderungen am System vorgenommen hat), bekommt man die Meldung: `The following packages have been kept back`. Abhilfe besteht dann in einem Voll-Upgrade:

Terminal
```
root@debian964:~# apt dist-upgrade
```

Mit `dist-upgrade` können auch neue Pakete installiert und alte gelöscht werden, was beim normalen `upgrade` niemals geschieht. `dist-upgrade` benutzt man meist nur beim Übergang von einer Debian-Version auf die nächste.

Mit `apt` gibt es außerdem folgende Möglichkeiten:

Terminal
```
root@debian964:~# apt install joe # installiere joe
                                  # von einer Paketquelle
root@debian964:~# apt install joe-3.6- joe-3.7+ # ersetze joe-3.6
                                  # durch joe-3.7
root@debian964:~# apt remove joe # lösche joe
root@debian964:~# apt purge joe  # lösche joe, auch Konfig.-Dateien
root@debian964:~# apt clean      # lösche /var/cache/apt/archives/*
root@debian964:~# apt autoclean  # dto.,aber nur,falls neuere Version
                                  # vorliegt
root@debian964:~# apt autoremove # lösche nicht mehr benötigte Pakete,
                                  # die nur wegen alter Abhängigkeiten
                                  # installiert worden waren
root@debian964:~# apt purge joe  # dto., auch Konfigurations-Dateien
```

Die speziellen Programme `apt-cache`, `apt-get`, `apt-cdrom` können noch weitere Einzelheiten abdecken, z. B. :

Terminal
```
root@debian964:~# apt-cdrom add # füge Wechseldatenträger als
                                 # Paketquelle hinzu
```