Debian-Packaging-Tutorial

David Frey

Motivation — Wieso packagen? (1)

Eine häufige Frage ist “*Wieso packagen? Ich kann doch gleich* ‘make install’ *benutzen?*”.

(Binary-)**Packaging** hat folgende Ziele:

1. Das Paket soll *verteilbar* und auf anderen Systemen *installierbar* sein.
   $\Rightarrow$ Die *Abhängigkeiten* des Pakets müssen *bekannt* sein.
2. Meta-Informationen für den Benutzer zu liefern (**dpkg -I**):
   - was macht das Paket?
   - welche anderen Pakete haben mit diesem einen Zusammenhang?
   - Grösse desselben?
   - Welche Files sind Konfigurationsfiles?

Motivation — Wieso packagen? (2)

Zur erfolgreichen Installation (z. B. mit **dpkg -i**) sind folgende Informationen nötig:

1. *Zustand* des zu installierenden Pakets (**dpkg -s**):
   { unknown, install, remove, purge, hold }
2. Zustand der anderen Pakete (**dpkg -l**):
   - Was ist schon installiert?
   - Welchen Zustand haben die anderen Pakete?
3. Was sind die Abhängigkeiten des gerade zu installierenden Pakets?
4. Kollidiert das Paket mit einem schon installiertem?

Motivation — Wieso packagen? (3)

Zusätzliche Informationen, die dem Paket-Manager (resp. dessen Front-End) bekannt sind:

1. Welche Pakete sind in welcher Distribution verfügbar?
2. Welche Pakete sind funktionell äquivalent? ⇒Virtual Packages
3. Welche Pakete sind *essentiell* für das Funktionieren des System?
4. Klassifikation des Pakets

Diese Information wird bei Systemupgrades, Plausibilitätschecks und bei der Darstellung der Paketlisten verwendet.

Aufbau eines `.deb`-Pakets

Debian-Pakete sind (vereinfachte) **ar**-Archive und können mit **ar** dargestellt und extrahiert werden.[a] Konstruiert werden sie allerdings von **dpkg-deb** [1], resp. dem Front-End dazu, **dpkg** [2].

```
$ar -t magicfilter_1.2-55_i386.deb
debian-binary
control.tar.gz
data.tar.gz
```

1. `debian-binary` enthält Versionierungsinformationen.
2. `control.tar.gz` enthält
   - eventuelle Checksummen,
   - die Installationsskripts (`prerm`, `postrm`, `preinst`, `postinst`),
   - die Configurationsfiles (`conffiles`),
   - und das control-File (`control`).
3. `data.tar.gz` sind die Daten als komprimiertes Tar-Archiv.

[a] Dies wird allerdings nur auf Plattformen gemacht, die kein installiertes **dpkg** besitzen, z. B. Sun Solaris

Files im `debian`-Verzeichnis

```
$ls debian/
changelog  copyright  postinst*  prerm*  substvars
control    menu       postrm*    rules*
```

- Installationsskripts (`prerm`, `postrm`, `postinst`)
- das `control`-File (`control`),
- das `changelog` und `copyright`-File, landet schlussendlich unter `/usr/share/doc/$PACKAGE`,
- das `rules`-**Makefile**, das macht die ganze Arbeit
- das `menu`-File für das Menü-System. Beschreibt den Menü-Eintrag (optional, falls sinnvoll)
- das `doc-base`-File für die Dokumentenregistrierung. Trägt das Dokument mit **install-docs** in die Debian-Dokumentenverwaltung ein (optional, falls sinnvoll)

debian/control (1)

Das control-File [3] beinhaltet die *Metadaten:*

control

```
Source: magicfilter
Build-Depends: dvips, netpbm|pbmplus, pnmtopng,
 libjpeg-progs|libjpeg-gif, libtiff-tools, groff,
 tetex-bin, transfig, recode, djtools, gs,
 enscript, bzip2, gzip (>= 1.2.4-33),
 xpdf-utils
Section: text
Priority: optional
Maintainer: David Frey <dfrey@debian.org>
Standards-Version: 3.5.5

Package: magicfilter
Architecture: any
Depends: $shlibs:Depends, libpaperg, enscript
Recommends: lpr|lprng, gs, bzip2,
 gzip (>= 1.2.4-33), xpdf-utils
Suggests: dvips, pbmplus|netpbm, pnmtopng,
 libjpeg-progs|libjpeg-gif, libtiff-tools, groff,
 tetex-bin, transfig, recode, djtools
Conflicts: apsfilter, printfilters-ppd
Description: automatic printer filter
...
```

`debian/control` (2)

- Welche Achitektur? → `Architecture:`
- Welche Libraries wurden zum Compilieren in welchen Versionen verwendet?

  → `Depends: ${shlibs:Depends}`
- Welche Interpreter/Programme werden in welcher Version verwendet und müssen zwingend vorhanden sein?

  → `Depends: enscript`
- Welche andere Programme werden im Normalfall verwendet (man kann die Programme weglassen, wenn man weiss, was man macht, z. B. bei magicfilter Einsatz eines PostScript-Druckers oder es wird nicht über lpr gedruckt)?

  → `Recommends:`
- Welche andere Programme werden potentiell aufgerufen und sind vorteilhaft installiert? In welchen Versionen?

  → `Suggests:`

`debian/{post,pre}{inst,rm}` — Installationsskripts

Die Installationsscripts machen die ganze Verwaltungsarbeit, z.B. das Paket beim Menu/Dokumenten-System registrieren.

postinst

```
#!/bin/sh -e
# -*- shell-script -*-

if test -x /usr/bin/update-menus; then
  update-menus
fi

cat<<EOF
Run \`magicfilterconfig' if you want to have
your printcap automatically generated or edit
it by hand.
EOF
```

Das Wichtigste an den Skripts ist, dass sie **idempotent** (beliebig oft erfolgreich aufrufbar).

Input kann interaktiv von `/dev/tty` kommen, oder, mittlerweile bevorzugt, über **debconf**.

`debian/rules` (1)

`debian/rules` ist ein **Makefile**, das dass Paket tatsächlich baut.

Zuerst ein paar Variablendefinitionen:

rules

```
#!/usr/bin/make -f
# -*- makefile -*-

# The name of the package
package := $(shell\
sed -n -e 's/Package: *\([a-z]\)/\1/p' \
debian/control)
doc     := usr/share/doc/$(package)
man     := usr/share/man/man8
menu    := usr/lib/menu
tmp     := debian/tmp

CFLAGS = -O2 -Wall
LDFLAGS=

ifneq (,$(findstring debug,$(DEB_BUILD_OPTIONS)))
  CFLAGS += -g
else
  LDFLAGS += -Wl,-s
endif
```

`debian/rules` (2)

Wie baut man dieses Package?

rules

```
build:
        ./configure --prefix=/usr \
                    --infodir=/usr/share/info \
                    --mandir=/usr/share/man \
                    --bindir=/usr/sbin && \
        make CFLAGS="$(CFLAGS)" \
             LDFLAGS="$(LDFLAGS)" \
             bindir=/usr/sbin && \
        touch build
```

Wie räumt man es wieder auf?

rules

```
clean:
        -rm -f build
        -$(MAKE) -i distclean || \
         $(MAKE) -f Makefile.in distclean
        -rm -rf *~ filters/*~ \
                $(tmp) debian/*~ \
                debian/files*
        -rm -rf debian/tmp
```

→

`debian/rules` (3)

Jedes `debian/rules` hat 2 Targets:
binary-indep für architekturneutrale Sachen (typischerweise Dokumentation) und
binary-arch für architekturabhängige Dinge.

rules

```
binary-indep: build
# There are no architecture-independent files
# to be uploaded generated by this package.
# If there were any they would be made here.

binary-arch: build
        -rm -rf $(tmp)
        install -d $(tmp)
        cd $(tmp) &&\
          install -d usr/sbin $(doc) $(man) \
          $(menu) etc/magicfilter
        $(MAKE) install prefix=$(tmp)/usr \
          bindir=$(tmp)/usr/sbin \
          infodir=$(tmp)/$(info) \
          mandir=$(tmp)/$(man)
```

→

debian/rules (4)

rules

```
ifeq (,$(findstring nostrip,$(DEB_BUILD_OPTIONS)))
        strip --remove-section=.comment \
              --remove-section=.note \
              --strip-unneeded \
              $(tmp)/usr/sbin/magicfilter
endif
        install magicfilterconfig \
           $(tmp)/usr/sbin/magicfilterconfig
        install -m 644 filters/README* \
          $(tmp)/$(doc)/
        install -m 644 debian/menu \
          $(tmp)/$(menu)/$(package)
        cd filters && \
          cp -p *-filter ../$(tmp)/etc/magicfilter
        cp debian/changelog \
          $(tmp)/$(doc)/changelog.Debian
        cd $(tmp)/$(doc) && \
          gzip -9f changelog.Debian README*
        cp debian/copyright $(tmp)/$(doc)
        cp magicfilterconfig.8 $(tmp)/$(man)
        gzip -9f $(tmp)/$(man)/*
```

→

debian/rules (5)

Hier fängt der Debian-spezifische Directory/Packaging-Teil an:

rules

```
	dpkg-shlibdeps $(tmp)/usr/sbin/$(package)
		install -d $(tmp)/DEBIAN
		cd $(tmp) &&\
		  md5sum `find * -name DEBIAN -prune \
		   -o -type f -print`\
		   > DEBIAN/md5sums
		dpkg-gencontrol -isp
		install -m 755 debian/prerm \
		               debian/postrm \
		               debian/postinst \
		               $(tmp)/DEBIAN
		cd filters && \
		ls -1 *-filter|\
		sed -e 's,^,/etc/magicfilter/,'\
		  > ../$(tmp)/DEBIAN/conffiles
		chmod 644 $(tmp)/DEBIAN/conffiles
		chown -R root.root $(tmp)
		chmod -R g-ws,og=rX $(tmp)
		dpkg --build $(tmp) ..
```

`debian/rules` (6)

Folgende Targets werden von **dpkg-buildpackage** erwartet:

rules

```
        # Below here is fairly generic really

binary:         binary-indep binary-arch

source diff:
        @echo >&2 'source and diff are obsolete \
- use dpkg-source -b'; false

.PHONY: binary binary-arch binary-indep clean
```

Die Korrektheit bezüglich Policy kann mit **lintian** und **linda** überprüft werden.

Literatur

[1] Debian Project. *dpkg-deb.*
`dpkg-deb(1).`

[2] Debian Project. *dpkg.*
`dpkg(8).`

[3] Debian Project. *deb-control.*
`deb-control(5).`