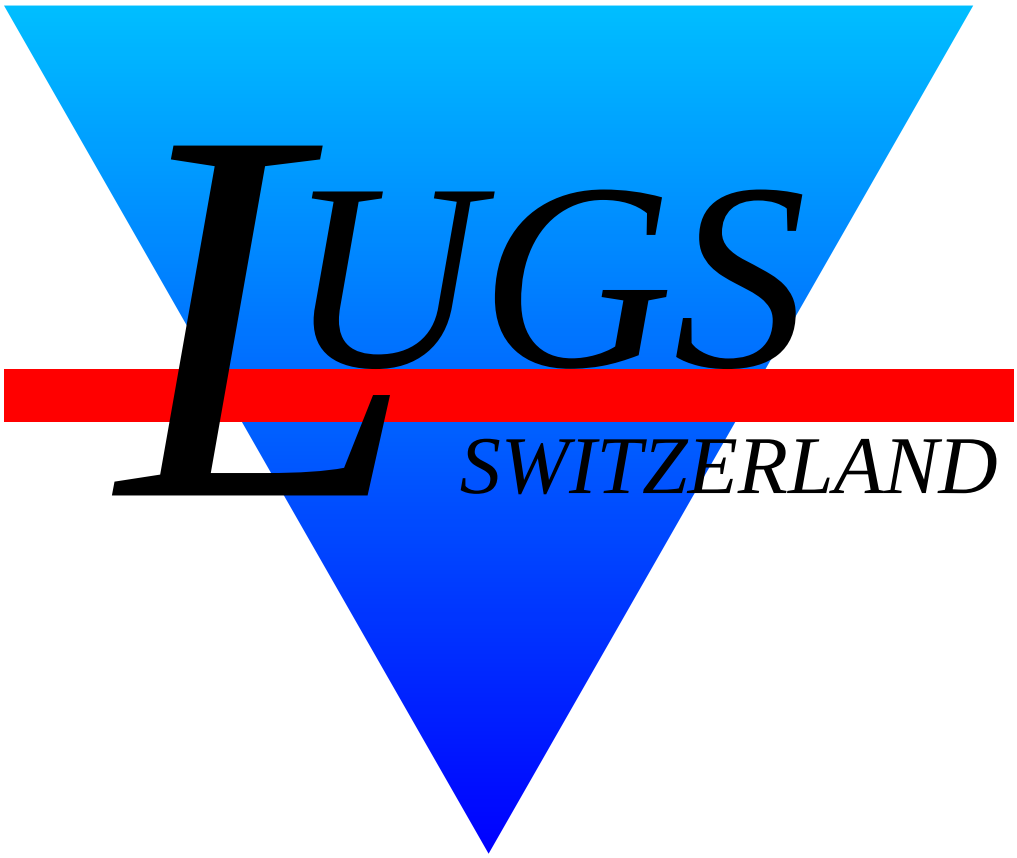
Debian-Packaging-Tutorial

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.
   ⇒ 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

```
1  #!/usr/bin/make -f
2  # -*- makefile -*-
3
4  # The name of the package
5  package := $(shell\
6  sed -n -e 's/Package: *\([a-z]\)/\1/p' \
7  debian/control)
8  doc     := usr/share/doc/$(package)
9  man     := usr/share/man/man8
10 menu    := usr/lib/menu
11 tmp     := debian/tmp
12
13 CFLAGS = -O2 -Wall
14 LDFLAGS=
15
16 ifneq (,$(findstring debug,$(DEB_BUILD_OPTIONS)))
17   CFLAGS += -g
18 else
19   LDFLAGS += -Wl,-s
20 endif
```

debian/rules (2)

Wie baut man dieses Package?

rules

```
21 build:
22         ./configure --prefix=/usr \
23                     --infodir=/usr/share/info \
24                     --mandir=/usr/share/man \
25                     --bindir=/usr/sbin && \
26         make CFLAGS="$(CFLAGS)" \
27              LDFLAGS="$(LDFLAGS)" \
28              bindir=/usr/sbin && \
29         touch build
```

Wie räumt man es wieder auf?

rules

```
30 clean:
31         -rm -f build
32         -$(MAKE) -i distclean || \
33          $(MAKE) -f Makefile.in distclean
34         -rm -rf *~ filters/*~ \
35                 $(tmp) debian/*~ \
36                 debian/files*
37         -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

```
38 binary-indep: build
39 # There are no architecture-independent files
40 # to be uploaded generated by this package.
41 # If there were any they would be made here.
42
43 binary-arch: build
44         -rm -rf $(tmp)
45         install -d $(tmp)
46         cd $(tmp) &&\
47           install -d usr/sbin $(doc) $(man) \
48           $(menu) etc/magicfilter
49         $(MAKE) install prefix=$(tmp)/usr \
50           bindir=$(tmp)/usr/sbin \
51           infodir=$(tmp)/$(info) \
52           mandir=$(tmp)/$(man)
```

→

debian/rules (4)

rules

```
53  ifeq (,$(findstring nostrip,$(DEB_BUILD_OPTIONS)))
54          strip --remove-section=.comment \
55                --remove-section=.note \
56                --strip-unneeded \
57                $(tmp)/usr/sbin/magicfilter
58  endif
59          install magicfilterconfig \
60             $(tmp)/usr/sbin/magicfilterconfig
61          install -m 644 filters/README* \
62            $(tmp)/$(doc)/
63          install -m 644 debian/menu \
64            $(tmp)/$(menu)/$(package)
65          cd filters && \
66            cp -p *-filter ../$(tmp)/etc/magicfilter
67          cp debian/changelog \
68            $(tmp)/$(doc)/changelog.Debian
69          cd $(tmp)/$(doc) && \
70            gzip -9f changelog.Debian README*
71          cp debian/copyright $(tmp)/$(doc)
72          cp magicfilterconfig.8 $(tmp)/$(man)
73          gzip -9f $(tmp)/$(man)/*
```

→

`debian/rules` (5)

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

rules

```
74      dpkg-shlibdeps $(tmp)/usr/sbin/$(package)
75          install -d $(tmp)/DEBIAN
76          cd $(tmp) &&\
77            md5sum `find * -name DEBIAN -prune \
78             -o -type f -print`\
79             > DEBIAN/md5sums
80          dpkg-gencontrol -isp
81          install -m 755 debian/prerm \
82                         debian/postrm \
83                         debian/postinst \
84                         $(tmp)/DEBIAN
85          cd filters && \
86          ls -1 *-filter|\
87          sed -e 's,^,/etc/magicfilter/,'\
88            > ../$(tmp)/DEBIAN/conffiles
89          chmod 644 $(tmp)/DEBIAN/conffiles
90          chown -R root.root $(tmp)
91          chmod -R g-ws,og=rX $(tmp)
92          dpkg --build $(tmp) ..
```

→

`debian/rules` (6)

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

rules

```
93      # Below here is fairly generic really
94
95  binary:         binary-indep binary-arch
96
97  source diff:
98          @echo >&2 'source and diff are obsolete \
99  - use dpkg-source -b'; false
100
101 .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)`.