Debian Paketbau - Technische Einführung

Carsten Schönert

10. November 2014

Themenübersicht

1. Was ist ein Debian Paket?
2. Im Detail
3. Vorbereitungen
4. Praxis am Beispiel
5. Weitere Infoquellen

Allgemeines

Es gibt zwei Typen Debian Pakete, aber ...
im Allgemeinen werden **Binärpakete** gemeint ($\Rightarrow$ *.deb).

- enthalten Programme (Binarys, Interpretercode), Konfigurationen, Manpages, Dokumentationen, Development-Dateien, Copyright Infos
- aber auch Steuerinfos für Paketmanager (apt, aptitude, dpkg, gdebi etc.)
- mit 'ar' entpackbares Archiv (Vorläufer von 'tar', wird auch benutzt um statische Libs zu erzeugen)
- manche sind Plattform spezifisch (z.B. i386, amd64, powerpc, armel, arm64 etc.)

Allgemeines

Es gibt zwei Typen Debian Pakete, aber ...
im Allgemeinen werden **Binärpakete** gemeint ($\Rightarrow$ *.deb).

- enthalten Programme (Binarys, Interpretercode), Konfigurationen, Manpages, Dokumentationen, Development-Dateien, Copyright Infos
- aber auch Steuerinfos für Paketmanager (apt, aptitude, dpkg, gdebi etc.)
- mit 'ar' entpackbares Archiv (Vorläufer von 'tar', wird auch benutzt um statische Libs zu erzeugen)
- manche sind Plattform spezifisch (z.B. i386, amd64, powerpc, armel, arm64 etc.)

Es gibt aber auch noch **Quellpakete** (*.dsc und *.orig.tar.gz)

- keine direkt installierbare Dateien, dienen zum Erstellen der Binärpakete

Aufbau im Detail

Inhalt einer *.deb Datei (ar -t foo.deb)

debian-binary
control.tar.gz
data.tar.{bz2,gz,lzma,xz}

Aufbau im Detail

Inhalt einer *.deb Datei (ar -t foo.deb)

debian-binary
control.tar.gz
data.tar.{bz2,gz,lzma,xz}

debian-binary:

- Textdatei mit Versionsnummer, aktuell 2.0

control.tar.gz:

- enthält Dateien die zur Installation dienen oder Abhängigkeiten auflisten sowie Prüfsummen

data.tar.{bz2,gz,lzma,xz}

- enthält die eigentlichen Programmdaten

Was benötigt man für Paketbau?

Im Allgemeinen:

- grundlegende Erfahrung über Funktionsweise eines Linuxsystems (wie z.B. LSB *Linux Standard Base*, FHS *File Hierachy Standard*)

Was benötigt man für Paketbau?

Im Allgemeinen:

- grundlegende Erfahrung über Funktionsweise eines Linuxsystems (wie z.B. LSB *Linux Standard Base*, FHS *File Hierachy Standard*)
- Erfahrung mit der Programmiersprache des zu paketierenden Programm

Was benötigt man für Paketbau?

Im Allgemeinen:

- grundlegende Erfahrung über Funktionsweise eines Linuxsystems (wie z.B. LSB *Linux Standard Base*, FHS *File Hierachy Standard*)
- Erfahrung mit der Programmiersprache des zu paketierenden Programm
- Fortgeschritten: Know How verschiedene Desktop Environments, Init Systeme, Besonderheiten bei Versionsupdates, Lizenzen, ...

Was benötigt man für Paketbau?

Im Allgemeinen:

- grundlegende Erfahrung über Funktionsweise eines Linuxsystems (wie z.B. LSB *Linux Standard Base*, FHS *File Hierachy Standard*)
- Erfahrung mit der Programmiersprache des zu paketierenden Programm
- Fortgeschritten: Know How verschiedene Desktop Environments, Init Systeme, Besonderheiten bei Versionsupdates, Lizenzen, ...
- **Interesse und Geduld, Geduld, Geduld!** ☺

Was benötigt man im Detail?

Im Speziellen:

1. Devel Pakete installieren

```
$ sudo apt-get update
$ sudo apt-get install build-essential devscripts dh-make
```

2. Source holen und entpacken, originalen Tarball in `$package_$version.orig.tar.*` umbenennen
3. Verzeichnis `debian/` erstellen und benötigte Steuerdateien anlegen

Noch etwas Theorie ... benötigte Dateien in debian/

Datei	Zweck

Noch etwas Theorie ... benötigte Dateien in debian/

Datei	Zweck
control	Steuerdatei für Buildtools

Noch etwas Theorie ... benötigte Dateien in debian/

Datei	Zweck
control	Steuerdatei für Buildtools
rules	Regelwerk zur Paketerstellung (Makefile!)

Noch etwas Theorie ... benötigte Dateien in debian/

Datei	Zweck
control	Steuerdatei für Buildtools
rules	Regelwerk zur Paketerstellung (Makefile!)
changelog	wie der Name schon sagt ...

Noch etwas Theorie ... benötigte Dateien in debian/

Datei	Zweck
control	Steuerdatei für Buildtools
rules	Regelwerk zur Paketerstellung (Makefile!)
changelog	wie der Name schon sagt ...
copyright	wie der Name schon sagt ...

Noch etwas Theorie ... benötigte Dateien in debian/

Datei	Zweck
control	Steuerdatei für Buildtools
rules	Regelwerk zur Paketerstellung (Makefile!)
changelog	wie der Name schon sagt ...
copyright	wie der Name schon sagt ...
compat	Kompatibilitätslevel für debhelper

Noch etwas Theorie ... benötigte Dateien in debian/

Datei	Zweck
control	Steuerdatei für Buildtools
rules	Regelwerk zur Paketerstellung (Makefile!)
changelog	wie der Name schon sagt ...
copyright	wie der Name schon sagt ...
compat	Kompatibilitätslevel für debhelper
source/format	Version für das Format vom Quellpaket

Tabelle: minimal sechs benötigte Dateien in debian/

Nun aber etwas Praxis!

Paketerstellung am Beispiel von bgs

Ein paar nötige Variablen setzen:

```
$ grep DEB ~/.bashrc
DEBEMAIL=c.schoenert@t-online.de
DEBFULLNAME="Carsten␣Schoenert"
export DEBEMAIL DEBFULLNAME
```

Quelle: https://github.com/Gottox/bgs
RFP Bug: https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=745451

Paketerstellung am Beispiel von bgs

Ein paar nötige Variablen setzen:

```
$ grep DEB ~/.bashrc
DEBEMAIL=c.schoenert@t-online.de
DEBFULLNAME="Carsten␣Schoenert"
export DEBEMAIL DEBFULLNAME
```

Quelle: https://github.com/Gottox/bgs
RFP Bug: https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=745451

```
$ wget https://github.com/Gottox/bgs/releases/\
download/0.6/bgs-0.6.tar.gz
$ mv bgs-0.6.tar.gz bgs_0.6.orig.tar.gz
$ tar xvzf bgs_0.6.orig.tar.gz
$ cd bgs-0.6
$ dh_make -s
```

Was macht dh-make?

```
$ ls -la debian/
insgesamt 112
drwxr-xr-x 3 carsten carsten  4096 Nov  5 19:33 .
drwxr-xr-x 3 carsten carsten  4096 Nov  4 21:50 ..
-rw-r--r-- 1 carsten carsten   119 Nov  4 19:30 bgs.cron.d.ex
-rw-r--r-- 1 carsten carsten   223 Nov  4 19:30 bgs.default.ex
-rw-r--r-- 1 carsten carsten   471 Nov  4 19:30 bgs.doc-base.EX
-rw-r--r-- 1 carsten carsten   191 Nov  4 19:30 changelog
-rw-r--r-- 1 carsten carsten     2 Nov  4 19:30 compat
-rw-r--r-- 1 carsten carsten   514 Nov  4 19:30 control
-rw-r--r-- 1 carsten carsten  1679 Nov  4 19:30 copyright
...
-rw-r--r-- 1 carsten carsten   926 Nov  4 19:30 postrm.ex
-rw-r--r-- 1 carsten carsten   686 Nov  4 19:30 preinst.ex
-rw-r--r-- 1 carsten carsten   873 Nov  4 19:30 prerm.ex
-rw-r--r-- 1 carsten carsten   181 Nov  4 19:30 README.Debian
-rw-r--r-- 1 carsten carsten   266 Nov  4 19:30 README.source
-rwxr-xr-x 1 carsten carsten   914 Nov  4 21:47 rules
drwxr-xr-x 2 carsten carsten  4096 Nov  4 19:30 source
-rw-r--r-- 1 carsten carsten   757 Nov  4 19:30 watch.ex
```

Na dann mal los! Oder?

Im Prinzip können wir ja nun schon loslegen!

Na dann mal los! Oder?

Im Prinzip können wir ja nun schon loslegen!

```
$ debuild -us -uc
```

Na dann mal los! Oder?

Im Prinzip können wir ja nun schon loslegen!

```
$ debuild -us -uc
```

Ein Buildfehler!

```
make[1]: Entering directory '/tmp/buildd/bgs-0.6'
bgs build options:
CFLAGS   = -std=c99 -pedantic -Wall -Os -I. -I/usr/include -I/usr/
LDFLAGS  = -s -L/usr/lib -lc -L/usr/X11R6/lib -lX11 -L/usr/X11R6/l
CC       = cc
CC bgs.c
bgs.c:7:22: fatal error: X11/Xlib.h: No such file or directory
 #include <X11/Xlib.h>
                      ^
compilation terminated.
Makefile:18: recipe for target 'bgs.o' failed
```

Da fehlt ein Header, vermutlich noch mehr.

Buildfehler ...

Buildfehler sind beim ersten Paketieren normal. **Don't Panic!**
Bei diesem Paket hilft ein Blick in die Buildflags (config.mk). Wir müssen ein paar *-dev Pakete nachinstallieren, die die benötigten Header beinhalten. Und diese Pakete in die Build Abhängigkeiten aufnehmen. Die geschieht in der Datei debian/control.

Buildfehler ...

Buildfehler sind beim ersten Paketieren normal. **Don't Panic!**
Bei diesem Paket hilft ein Blick in die Buildflags (config.mk). Wir müssen ein paar *-dev Pakete nachinstallieren, die die benötigten Header beinhalten. Und diese Pakete in die Build Abhängigkeiten aufnehmen. Die geschieht in der Datei debian/control.

```
$ sudo apt-get install libimlib2-dev libx11-dev libxinerama-dev
...
$ vim debian/control
...
-Build-Depends: debhelper (>= 9)
+Build-Depends: debhelper (>= 9),
+  libimlib2-dev,
+  libx11-dev,
+  libxinerama-dev
```

Neuer Buildversuch ...

Buildfehler ...

Hmm.... ein anderer Fehler!

```
installing executable file to .../bgs-0.6/debian/bgs/usr/local/bin
installing manual page to .../bgs-0.6/debian/bgs/usr/local/share/m
make[1]: Leaving directory '.../bgs-0.6'
   dh_installdocs
   dh_installchangelogs
   dh_perl
   dh_usrlocal
dh_usrlocal: debian/bgs/usr/local/share/man/man1/bgs.1 is not a di
rmdir: konnte 'debian/bgs/usr/local/share/man/man1' nicht entferne
```

Buildfehler ...

Hmm.... ein anderer Fehler!

```
installing executable file to .../bgs-0.6/debian/bgs/usr/local/bin
installing manual page to .../bgs-0.6/debian/bgs/usr/local/share/m
make[1]: Leaving directory '.../bgs-0.6'
   dh_installdocs
   dh_installchangelogs
   dh_perl
   dh_usrlocal
dh_usrlocal: debian/bgs/usr/local/share/man/man1/bgs.1 is not a di
rmdir: konnte 'debian/bgs/usr/local/share/man/man1' nicht entferne
```

Der PREFIX für die Installation stimmt nicht! Können wir in debian/rules anpassen indem wir ein Override benutzen. Solche Overrides geben uns Flexibilität um entweder Besonderheiten von Upstream zu 'umschiffen' oder Debian Spezifika einzuhalten.

```
override_dh_auto_install:
        $(MAKE) DESTDIR=$$(pwd) PREFIX=usr/ install
```

Und nun?

Was nun? War das jetzt alles?

Nicht ganz! Nun kommt der genauso wichtige Teil das wir die Qualität unseres Paketes sicher stellen müssen.

Und nun?

Was nun? War das jetzt alles?

Nicht ganz! Nun kommt der genauso wichtige Teil das wir die Qualität unseres Paketes sicher stellen müssen.

- Lässt sich unser Paket erfolgreich installieren **und** benutzen?

Und nun?

Was nun? War das jetzt alles?

Nicht ganz! Nun kommt der genauso wichtige Teil das wir die Qualität unseres Paketes sicher stellen müssen.

- Lässt sich unser Paket erfolgreich installieren **und** benutzen?
- Wenn wir Plattform spezifischen Code erstellen, wie sieht die Buildfähigkeit auf anderen CPU Architekturen aus?

Und nun?

Was nun? War das jetzt alles?

Nicht ganz! Nun kommt der genauso wichtige Teil das wir die Qualität unseres Paketes sicher stellen müssen.

- Lässt sich unser Paket erfolgreich installieren **und** benutzen?
- Wenn wir Plattform spezifischen Code erstellen, wie sieht die Buildfähigkeit auf anderen CPU Architekturen aus?
- Haben wir die Debian Richtlinien eingehalten? (Lintian)

Und nun?

Was nun? War das jetzt alles?

Nicht ganz! Nun kommt der genauso wichtige Teil das wir die Qualität unseres Paketes sicher stellen müssen.

- Lässt sich unser Paket erfolgreich installieren **und** benutzen?
- Wenn wir Plattform spezifischen Code erstellen, wie sieht die Buildfähigkeit auf anderen CPU Architekturen aus?
- Haben wir die Debian Richtlinien eingehalten? (Lintian)
- Benötigen wir eine Kommunikation mit dem Benutzer während der Installation/Update? (debconf)

Und nun?

Was nun? War das jetzt alles?

Nicht ganz! Nun kommt der genauso wichtige Teil das wir die Qualität unseres Paketes sicher stellen müssen.

- Lässt sich unser Paket erfolgreich installieren **und** benutzen?
- Wenn wir Plattform spezifischen Code erstellen, wie sieht die Buildfähigkeit auf anderen CPU Architekturen aus?
- Haben wir die Debian Richtlinien eingehalten? (Lintian)
- Benötigen wir eine Kommunikation mit dem Benutzer während der Installation/Update? (debconf)
- Welche Abhängigkeiten, oder auch Konflikte mit anderen Paketen gibt es?

Wo gibt es weitere Infos, Anleitungen? Wie gehts weiter?

Es gibt zahlreiche Webseiten, Howtos und Infos rund zum Thema Debian Paketbau.

- https://wiki.debian.org/IntroDebianPackaging

Wo gibt es weitere Infos, Anleitungen? Wie gehts weiter?

Es gibt zahlreiche Webseiten, Howtos und Infos rund zum Thema Debian Paketbau.

- https://wiki.debian.org/IntroDebianPackaging
- https://wiki.debian.org/CategoryPackaging

Wo gibt es weitere Infos, Anleitungen? Wie gehts weiter?

Es gibt zahlreiche Webseiten, Howtos und Infos rund zum Thema Debian Paketbau.

- https://wiki.debian.org/IntroDebianPackaging
- https://wiki.debian.org/CategoryPackaging
- http://wiki.ubuntuusers.de/Grundlagen_der_Paketerstellung

Wo gibt es weitere Infos, Anleitungen? Wie gehts weiter?

Es gibt zahlreiche Webseiten, Howtos und Infos rund zum Thema Debian Paketbau.

- https://wiki.debian.org/IntroDebianPackaging
- https://wiki.debian.org/CategoryPackaging
- http://wiki.ubuntuusers.de/Grundlagen_der_Paketerstellung
- http://mentors.debian.net/

Wo gibt es weitere Infos, Anleitungen? Wie gehts weiter?

Es gibt zahlreiche Webseiten, Howtos und Infos rund zum Thema Debian Paketbau.

- https://wiki.debian.org/IntroDebianPackaging
- https://wiki.debian.org/CategoryPackaging
- http://wiki.ubuntuusers.de/Grundlagen_der_Paketerstellung
- http://mentors.debian.net/
- im IRC #debian-mentors (#debian-devel) auf irc.oftc.net

Wo gibt es weitere Infos, Anleitungen? Wie gehts weiter?

Es gibt zahlreiche Webseiten, Howtos und Infos rund zum Thema Debian Paketbau.

- https://wiki.debian.org/IntroDebianPackaging
- https://wiki.debian.org/CategoryPackaging
- http://wiki.ubuntuusers.de/Grundlagen_der_Paketerstellung
- http://mentors.debian.net/
- im IRC #debian-mentors (#debian-devel) auf irc.oftc.net
- Abschauen ist explizit erlaubt. Ist alles Open Source! ☺
  z.B. über
  https://packages.qa.debian.org/common/index.html

Ich habe immer noch ein paar Fragen.

- Was ist dieses 'debhelper'? (CDBS vs. dh)

Ich habe immer noch ein paar Fragen.

- Was ist dieses 'debhelper'? (CDBS vs. dh)
- Wie und wozu benutze ich Chroot's? (Vorteile?, Paketbau für andere Releases $\Rightarrow$ i386 auf amd64?)

Ich habe immer noch ein paar Fragen.

- Was ist dieses 'debhelper'? (CDBS vs. dh)
- Wie und wozu benutze ich Chroot's? (Vorteile?, Paketbau für andere Releases $\Rightarrow$ i386 auf amd64?)
- Wie erstelle und benutze ich diese? (pbuilder, sbuild)

Ich habe immer noch ein paar Fragen.

- Was ist dieses 'debhelper'? (CDBS vs. dh)
- Wie und wozu benutze ich Chroot's? (Vorteile?, Paketbau für andere Releases $\Rightarrow$ i386 auf amd64?)
- Wie erstelle und benutze ich diese? (pbuilder, sbuild)
- Wie verwalte ich verschiedene Versionsstände des debian/ Verzeichnis? Und wozu? (git-buildpackage, git-dpm, hg-buildpackage, ...)

Ich habe immer noch ein paar Fragen.

- Was ist dieses 'debhelper'? (CDBS vs. dh)
- Wie und wozu benutze ich Chroot's? (Vorteile?, Paketbau für andere Releases $\Rightarrow$ i386 auf amd64?)
- Wie erstelle und benutze ich diese? (pbuilder, sbuild)
- Wie verwalte ich verschiedene Versionsstände des debian/ Verzeichnis? Und wozu? (git-buildpackage, git-dpm, hg-buildpackage, ...)
- Wie gehe ich mit Unittests um?

Ich habe immer noch ein paar Fragen.

- Was ist dieses 'debhelper'? (CDBS vs. dh)
- Wie und wozu benutze ich Chroot's? (Vorteile?, Paketbau für andere Releases $\Rightarrow$ i386 auf amd64?)
- Wie erstelle und benutze ich diese? (pbuilder, sbuild)
- Wie verwalte ich verschiedene Versionsstände des debian/ Verzeichnis? Und wozu? (git-buildpackage, git-dpm, hg-buildpackage, ...)
- Wie gehe ich mit Unittests um?

Zahlreiche weitere Fragen und Probleme.
Aber Rom wurde auch nicht an einem Tag erbaut!

Fragen?

Fragen? Gerne auch zu einem speziellen Paket!

Interesse an 'mehr' Debian Paketbau?

Danke für die Aufmerksamkeit!

©2014 Carsten Schönert c.schoenert [at] t-online [dot] de
Vielen Dank an Guido und Sascha für die Unterstützung!