I deb, you deb, everybody debs

Debian packaging for beginners and experts alike

Ondřej Surý  •  ondrej@isc.org  •  ondrej@debian.org  •  25.- 10. 2017
Contents

- .deb binary package structure
- Source package structure
- Basic toolchain
- Recommended toolchain
- Keeping the sources in git
- Clean build environment
- Misc...
- So how do I become Debian Developer?
My Debian portfolio (since 2000)

- Mostly team maintained
- PHP + PECL (pkg-php)
  - Co-installable packages since 7.x
  - Longest serving PHP maintainer in Debian (and still not crazy)
- libjpeg-turbo
  - Transitioned Debian from IIJ JPEG (that crazy guy) to libjpeg-turbo
- DNS Packaging Group
  - CZ.NIC’s Knot DNS and Knot Resolver
  - NLnet Lab’s NSD, Unbound, getdns, ldns
  - PowerDNS
  - BIND 9
- Berkeley DB and LMDB (pkg-db)
  - One Berkeley DB per release (yay!)
- BIRD
- Cyrus SASL
- Cyrus IMAPD
- Apache2 + mod_md (fresh)
- ...other little stuff

Older work

- GTK/GNOME/Freedesktop
- Redmine/Ruby
  - Never again, it’s a straight road to madness
Binary package structure

- **ar archive consisting of:**
  - **debian-binary**
    - .deb format version (2.0)
  - **control.tar.gz**
    - Package information (control)
    - Maintainer scripts
      - {pre,post}{inst,rm}
    - Misc (md5sum, conffiles)
  - **data.tar.xz**
    - Actual content of the package
    - This is what gets installed

- **Nástroje pro práci s .deb soubory**
  - dpkg-deb
  - dpkg (wrapper around dpkg-deb)
  - ar, tar

```
$ ar xv knot_2.0.1-4_amd64.deb
  x - debian-binary
  x - control.tar.gz
  x - data.tar.xz

$ dpkg-deb -X knot_2.0.1-4_amd64.deb output/
  ./
  ./etc/
  [...]
  ./usr/sbin/knotd
  [...]

$ dpkg-deb -e knot_2.0.1-4_amd64.deb DEBIAN/
$ ls DEBIAN/
  conffiles control md5sums postinst postrm preinst prerm

$ dpkg -I knot_2.0.1-4_amd64.deb
  new debian package, version 2.0.
  size 218134 bytes: control archive=2737 bytes.
  Package: knot
  Version: 2.0.1-4
  Architecture: amd64
```
Source package structure

- **Source package metadata**
  - `<pkg>_<dver>.dsc`
    - Maintainer + Uploaders
    - Build Dependencies and Conflicts
    - List of binary packages
    - Checksums (sha256)
    - And more...

- **Upstream tarball**
  - `<pkg>_<uver>.orig.?z`
    - Original upstream unmodified tarball
    - Or repacked due licensing reasons (dfsg)

- **Changes against the upstream tarball**
  - `<pkg>_<dver>.debian.tar.xz`
    - Debian directory (debian/)
    - Or other changes in older package formats
Debian package versioning

[epoch:]upstream_version[-debian_revision]

- **epoch** – [0-9]
- **upstream_version** – [0-9a-z.+-:~]
- **debian_version** – [0-9a-z.+~]

Version comparison
- First numerically compare **epoch**
- Then left to right **upstream_version**
  - Split to non-numeric and numeric parts
  - All letters goes before numbers
  - ~ loses even to an empty string
- Finally compare **debian_version**
  - Same algorithm as **upstream_version**

---

# fixup the versioning (start over)
# e.g. new package with same name or mistake
# example: git (formelly GNU Interactive Tools)
dpkg --compare-versions ‘1:2.1.4-2.1’ gt ‘4.3.20-7’

# repacked upstream tarball with non-free parts removed
dpkg --compare-versions ‘1.0.3-3’ lt ‘1.0.3+dfsg-1’

# botched upload without epoch bump
dpkg --compare-versions ‘2.1.0+really+2.0.0-1’ gt ‘2.1.0-1’

# final release trumps over alpha/beta/rc
dpkg --compare-versions ‘2.1.0-1’ gt ‘2.1.0-rc1-1’

# upload to Debian security or proposed-updates
dpkg --compare-versions ‘5.6.14-0+deb9u1’ lt ‘5.6.14-1’

# ~ vs “” examples
dpkg --compare-versions '0:0' lt '1' #true 0 is default
Dpkg --compare-versions '1:0' gt '1000' #true
dpkg --compare-versions '0~' lt '0' #true
dpkg --compare-versions '~~' lt '~~a' #true
dpkg --compare-versions '~~' lt '~' #true
Source package — debian/ directory

- **d/changelog**
  - List of **packaging** changes
  - aka Debian changelog
- **d/control**
  - Metadata for source and binary packages
- **d/rules**
  - Rules (Makefile) for building the package
- **d/copyright**
  - (Machine readable) list of source licenses
- **d/source/format**
  - **3.0 (quilt)** or **3.0 (native)**
- **d/patches/**
  - Patches for upstream sources
  - Managed by **quilt** or **gbp pq**
- And more...

```bash
$ ls -l d debian/*
debian/changelog
debian/compat
debian/control
debian/copyright
debian/docs
debian/init-d-script
debian/knot-resolver.default
debian/knot-resolver.init
debian/knot-resolver.install
debian/knot-resolver.lintian-overrides
debian/knot-resolvers.dirs
debian/knot-resolver.service
debian/knot-resolver.tmpfile
debian/kresd.conf
debian/patches/
debian/rules
debian/source/
```
Basic (barebone) toolchain

- **d/rules** is Makefile
- **d/rules <target>**, where <target> is:
  - **clean**
    - Return the pristine state
  - **build (build-arch, build-indep)**
    - Compile the sources
  - **install (install-indep, install-arch)**
    - Install into the temporary path
  - **binary (binary-arch, binary-indep)**
    - Assemble the file pages
- **Minimal d/rules →**
  - But don’t do this at home, kids

```bash
#!/usr/bin/make -f
# -*- makefile -*-
build:
  gcc -o helloworld helloworld.c

clean:
  rm -f helloworld

install: build
  mkdir debian/tmp/usr/bin
  cp -a helloworld debian/tmp/usr/bin/

binary-indep: build

binary-arch: build
  mkdir debian/tmp/DEBIAN/
  dpkg-gencontrol
dpkg-deb -b debian/tmp/ <package>_<dver>.deb

binary: binary-arch binary-indep

.PHONY: build clean install binary binary-arch binary-indep
```
dh_make – the first simple step

dh_make

● Creates a basic skeleton in debian/
● It’s not magic, you need to modify it
● Package types:
  ○ Single (--single)
    ■ Just single binary package
  ○ Multi
    ■ Multiple binary packages
  ○ Library
    ■ Shared library (libfoo0 + libfoo-dev)
  ○ …

dh_make_perl, gem2deb

● Simple CPAN or GEM packaging

$ tar -xJf knot-resolver-1.0-beta.tar.xz
$ cd knot-resolver-1.0-beta/
knot-resolver-1.0-beta$ dh_make -s -f \\n../knot-resolver-1.0-beta.tar.xz

Maintainer name : Ondřej Surý
Email-Address    : ondrej@debian.org
Date             : Fri, 09 Oct 2015 16:08:12 +0200
Package Name     : knot-resolver
Version          : 1.0-beta
License          : blank
Type of Package  : Single
Hit <enter> to confirm:
Done. Please edit the files in the debian/ subdirectory now. You should also check that the knot-resolver Makefiles install into $DESTDIR and not in /.
Recommended toolchain – debhelper

- Rich set of `dh_*` commands for packaging
  - `dh_auto_configure`, `dh_auto_build`, `dh_auto_install`, `dh_auto_test`
    - Automatic detection of build system (auto tool, cmake, and others)
  - `dh_install`, `dh_strip`, `dh_installcron`, ...
  - Compatibility levels (more features added)
    - v9 (d/compat)
      - Multi-arch support
      - Hardening (dpkg-buildflags)
    - v10
- Excensible framework (... in Perl)
- dh7 added dh command →
  - Runs a sequence of dh_* commands

```bash
#!/usr/bin/make -f
# See debhelper(7) (uncomment to enable)
# output every command that modifies files on the build system.
#DH_VERBOSE = 1

# see EXAMPLES in dpkg-buildflags(1) and read /usr/share/dpkg/*
DPKG_EXPORT_BUILDFLAGS = 1
include /usr/share/dpkg/default.mk

# see FEATURE AREAS in dpkg-buildflags(1)
export DEB_BUILD_MAINT_OPTIONS = hardening=+all

# see ENVIRONMENT in dpkg-buildflags(1)
# package maintainers to append CFLAGS
export DEB_CFLAGS_MAINT_APPEND = -Wall -pedantic
# package maintainers to append LDFLAGS
export DEB_LDFLAGS_MAINT_APPEND = -Wl,--as-needed

# main packaging script based on dh7 syntax
%
  dh $@ #DH7_ADDON#
```
Standard **debhelper** sequence

$ dh binary --no-act
dh_testdir
dh_auto_configure
dh_auto_build
dh_auto_test
dh_testroot
dh_prep
dh_installdirs
dh_auto_install
dh_install
dh_installdocs
dh_installchangelogs
dh_installexamples
dh_installman
dh_installcatalogs
dh_installcron
dh_installdebconf
dh_installmipsen
dh_installifupdown
dh_installinfo
dh_installinit
dh_installmenu
dh_installmime
dh_installmodules
dh_installlogcheck
dh_installlogrotate
dh_installpam
dh_installppp
dh_installudev
dh_installwm
dh_installsettings
dh_bugfiles
dh_ucf
dh_lintian
dh_gconf
dh_icons
dh_perl
dh_usrlocal
dh_link
dh_installxfonts
dh_compress
dh_fixperms
dh_strip
dh_make-shlibs
dh_shlibdeps
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
Files

**dh_installdirs**
- Helper script to create directories described in `d/<package>.dirs`

**dh_install**
- Helper script to copy `d/tmp/` contents into `d/<package>/` directories
- Described `d/<package>.install`
- It can’t rename files (doh!)
- Since compat level 9 the .install files can be executable (`#!/usr/bin/dh-exec`)
  - It can rename files
  - It can use env variables (f.e. Multi-Arch)

```bash
$ cat debian/knot-resolversdirs
/etc/knot-resolver
/var/lib/knot-resolver

$ cat debian/libknot1.install
usr/lib/*/libknot.so.*

$ cat debian/libknot-dev.install
usr/include/
usr/lib/*//*.a
usr/lib/*//*.so
usr/lib/*/pkgconfig/*

$ cat debian/knot-host.install
usr/bin/khost
usr/share/man/man1/khost.1

#!/usr/bin/dh-exec
src/libfoo-*.so.*
debian/foo-plugins/usr/lib/foo/{$DEB_HOST_MULTIARCH}/
etc/example.conf => debian/foo/etc/foo/foo.conf
[linux-any kfreebsd-any] arch-specific /usr/lib/foo/
```
dh_installdocs

- Helper script to install files from `d/<package>.docs + d/copyright, d/README.Debian` into `d/<package>/usr/share/doc/<package>`
- Option `--link-doc` can symlink documentation between packages
  - It can save space (in theory)
  - !!!WARNING!!! You must not mix binary:arch and binary:all packages, it break binNMU
  - My recommendation: DON'T USE IT
Tweak the standard dh behaviour

- **dh <target> --no-act**
  - Prints the chain of commands

- **Every command can be overridden in d/rules**
  - `override_<target>`

- **Some commands read env variables**

- **Some needs to be given option**
  - Via override
  - Or give it to `dh`

- **Common use:**
  - `dh_install`: --{list,fail}-missing option to loose/strict check of extra not-installed files
  - `dh_auto_configure` – pass extra ./configure options

```bash
$ dh build --no-act
dh_testdir
dh_auto_configure
dh_auto_build
debian/rules override_dh_auto_test

$ cat debian/rules
#!/usr/bin/make -f
[...
%:
  dh $@
# run tests, but don’t fail the build on failure
override_dh_auto_test:  
  -make check

# Example 2
$ cat debian/rules
%
  dh $@ --fail-missing
override_dh_auto_configure:  
  dh_auto_configure -- --enable-feature1
# --fail-missing gets passed to dh_install
```
Debhelper extensions

- Extensions for
  - Other build systems
    - PHP (PEAR, PECL)
  - More init systems
    - systemd (merged into dh10)
  - Different languages
    - PHP, Python, Ruby
  - Other stuff
    - autotools-dev (dh11)

#!/usr/bin/make -f

%:
  dh $@
  --with systemd
  --with autotools-dev
  --with autoreconf
  --with apache2
  --with python2
  --with python3
  ...

Some upstream tarballs are ancient or generated on Red^Hancient systems

- **dh_autoreconf**
  - Runs `autoreconf -fi`

- **dh_autotools-dev_{update,restore}config**
  - Updates `config.sub` and `config.guess` files
  - Sometimes needed to support new architectures (such as arm64)

- Both commands save modified files and restore the originals in the clean target

```bash
#!/usr/bin/make -f
%
  dh $@ --with autoreconf --with autotools_dev
override_dh_autoreconf:
  dh_autoreconf --as-needed
```
Debian Hardening

- There are several compiler options that can be used to harden the final binaries
  - Format security, Fortify source, Stack protector, PIE, relro, bindnow
- It might break the build though, so it's not enabled by default
- More in the Debian Wiki:
  https://wiki.debian.org/Hardening
  https://wiki.debian.org/HardeningWalkthrough

```bash
#!/usr/bin/make -f
DPKG_EXPORT_BUILDFLAGS = 1
include /usr/share/dpkg/default.mk
export DEB_BUILD_MAINT_OPTIONS = hardening=+all
%
   dh $@
```
Packaging with git (git-buildpackage)

**gbp [clone|import-dsc|import-org|buildpackage]**

- Set of tools to keep the packaging in the git
- It can also import existing source packages
- Upstream sources in the same git
- Debian packaging in a separate branch
- Common branches:
  - **upstream** – Upstream sources
  - **master** – Upstream sources + debian/
  - **pristine-tar** – delta to reconstruct orig.tar.?z
  - **master-stretch** – Branches for stable releases

```bash
$ cd knot
$ git describe
v2.0.1-86-gd04fab6
$ git archive \
  --output=/tmp/knot-2.0.1-86-gd04fab6.tar.xz \
  --prefix=knot-2.0.1-86-gd04fab6 HEAD
$ cd /tmp
$ gbp clone git://anonscm.debian.org/pkg-dns/knot.git
$ cd knot
$ gbp import-orig ../knot-2.0.1-86-gd04fab6.tar.xz
What is the upstream version? [86-gd04fab6] 2.0.1+86-gd04fab6
gbp:info: Importing '../knot-2.0.1-86-gd04fab6.tar.xz' to \n  branch 'upstream'...
gbp:info: Source package is knot
gbp:info: Upstream version is 2.0.1+86-gd04fab6
pristine-tar: committed
knot_2.0.1+86-gd04fab6.orig.tar.xz.delta to \n  branch pristine-tar
gbp:info: Merging to 'master'
gbp:info: Successfully imported version 2.0.1+86-gd04fab6 of \n  ../knot-2.0.1-86-gd04fab6.tar.xz
```
Lintian is your friend (and enemy)

`lintian <package>.changes`

- Checks built package according to the set of rules that evolves in a time
- It’s recommended to use stable-backports version
- Categories:
  - Errors – some will lead to reject on upload
  - Warnings - make sure you check them
  - Little things (lintian --pedantic)
- You can ignore false positives by listing the error in `d/<package>.lintian-overrides`

```bash
$ lintian --pedantic php5_5.6.14+dfsg-1_amd64.changes
P: php5 source: no-dep5-copyright
P: php5 source: debian-watch-may-check-gpg-signature
W: php5-common: binary-without-manpage /usr/sbin/php5dismod
W: php5-common: binary-without-manpage /usr/sbin/php5enmod
W: php5-common: binary-without-manpage /usr/sbin/php5query
W: php5-common: script-not-executable
  /usr/share/php5/php5-helper
W: php-pear: binary-without-manpage /usr/bin/pear
W: php-pear: binary-without-manpage /usr/bin/peardev
W: php-pear: binary-without-manpage /usr/bin/pecl
E: libapache2-mod-php5filter:
  apache2-module-depends-on-real-apache2-package apache2-bin
E: libapache2-mod-php5:
  apache2-module-depends-on-real-apache2-package apache2-bin
N: 33 tags overridden (12 errors, 7 warnings, 14 info)
```
How to contribute?

- Fix a package bug, package something, join an existing packaging team, write a documentation, translate package description into your language, ...
  - [https://www.debian.org/devel/join/newmaint](https://www.debian.org/devel/join/newmaint)
  - [http://mentors.debian.net/](http://mentors.debian.net/)

- Adopt an existing package or create a new
  - And become Debian Maintainer
    - [https://wiki.debian.org/DebianMaintainer](https://wiki.debian.org/DebianMaintainer)

- After you prove yourself and if you still like it
  - Became Debian Developer (with voting rights)
Library

- Debian Policy Manual
- Debian Developer’s Reference
- Debian New Maintainers' Guide
- Preferred debian packaging setup for dkg
  - https://wiki.debian.org/DanielKahnGillmor/preferred_packaging
Questions?