Debian Packaging

Theppitak Karoonboonyanan
thep@debian.org

Debian Developer

7 September 2011

Outline

Section 1

Preparation

Recommended Readings

- Debian New Maintainers' Guide
  - `sudo apt-get install maint-guide`
  - `http://www.debian.org/doc/manuals/maint-guide/`
- Debian Policy
  - `sudo apt-get install debian-policy`
  - `http://www.debian.org/doc/manuals/debian-policy/`
- Debian Developer's Reference
  - `sudo apt-get install developers-reference`
  - `http://www.debian.org/doc/manuals/developers-reference/`

Section 2

Building Packages

Getting Source

- Downloading
  - From Debian repository:

    ```
    apt-get source package
    ```

  - From FTP server:

    ```
    dget ftp://.../package_version.dsc
    ```

- Extracting

  ```
  dpkg-source -x package_version.dsc
  ```

Building Package

- Get build dependencies
  - For package in Debian repository:

```
sudo apt-get build-dep package
```

  - For package outside Debian repository:

```
mk-build-deps package_version.dsc
sudo dpkg -i package-build-deps_version.dsc
sudo apt-get -f install
```

- Build it

```
cd package-version
dpkg-buildpackage -rfakeroot
```

Multiple Levels of Package Building

- `dpkg-buildpackage`
  - Low level
  - Generates source, binary packages
  - Sign the `.dsc`
  - Generates changes file for upload
  - Sign the `.changes`
- `debuild`
  - `devscripts` wrapper for `dpkg-buildpackage`
  - Check `.changes` with `lintian`
  - Sign `.dsc` and `.changes` with `debsign`
- `pbuilder`
  - Build package in a clean base-system chroot
  - Any missing build-dependency will be caught
  - No link mistake against irrelevant libraries

dpkg-buildpackage

- Usage:

```
cd package-version
dpkg-buildpackage
```

- Interesting options:
  - `-k`*`key-id`* GPG key-ID used for signing
  - `-b` binary-only build
  - `-B` binary-only build, arc-dependent only
  - `-S` source-only build
  - `-sa` force `dpkg-genchanges` to include upstream source tarball
    (useful for mentors.debian.net)

debuild

- Usage:

```
cd package-version
debuild [--debbuildopts ...]
```

- Options for `dpkg-buildpackage` can be passed via `--debbuildopts` option
- Build log at `../*.build`

pbuilder

- Backends:
  - pbuilder (tgz)
  - cowbuilder (normal FS, copy-on-write)
  - qemubuilder (QEMU image)
- Create image:
  - `sudo pbuilder --create [--basetgz ...]`
  - `sudo cowbuilder --create [--basepath ...]`
  - `sudo qemubuilder --create --configfile ...`
- Update image:
  - `sudo pbuilder --update [--basetgz ...]`
  - `sudo cowbuilder --update [--basepath ...]`
  - `sudo qemubuilder --update --configfile ...`

pbuilder

- build:
  - `pbuilder --build` *`package-version`*`.dsc [--basetgz ...]`
  - `cowbuilder --build` *`package-version`*`.dsc [--basepath ...]`
  - `qemubuilder --build` *`package-version`*`.dsc --configfile ...`
- debuild:
  - `pdebuild [-- --basetgz ...]`
  - `pdebuild --pbuilder cowbuilder [-- --basepath ...]`
  - `pdebuild --pbuilder qemubuilder -- --configfile ...`

Section 3

Uploading Packages

Uploading Methods

- `dupload`
  - For pre-configured hosts in `/etc/dupload.conf` or `~/.dupload.conf`

```
dupload [-t host] package.changes
```

- `dput`
  - For pre-configured hosts in `/etc/dput.cf` or `~/.dput.cf`

```
dput [host] package.changes
```

- `scp`
  - For arbitrary hosts with SSH access

```
dcmd scp package.changes host:dir
```

- `cp`, `mv`, `rm`
  - For arbitrary dirs in local machine

```
dcmd cp package.changes dir
dcmd mv package.changes dir
dcmd rm package.changes dir
```

Section 4

Debian Packaging

Anatomy of Debian Source Package

- `debian/changelog`

```
package (1.0-1) unstable; urgency=low

  * Initial release. (Closes: #XXXXXX)

-- Joe Black <joe@abc.org> Fri, 26 Aug 2011 22:07:05 +0700
```

  - `package` = package name
  - `(1.0-1)` = version (*upstream-debian*)
  - `unstable` = target suite to upload to
  - `urgency=low` = migration urgency
  - `(Closes: #XXXXXX)` = Debian bug number this upload closes

Anatomy of Debian Source Package

- debian/copyright
  - upstream source copyright & license
  - debian packaging copyright & license
  - new format: DEP-5
    http://dep.debian.net/deps/dep5/

```
Format: http://svn.debian.org/wsvn/dep/web/deps/dep5.mdwn?op=fil
Upstream-Name: package
Upstream-Contact: Bill Parrish <bill@xyz.com>
Source: ftp://ftp.abc.com/pub/package/

Files: *
Copyright: 2011 Bill Parrish <bill@xyz.com>
License: GPL-2+
 On Debian GNU/Linux systems, the complete text of the GNU
 General Public License version 2 can be found in
 '/usr/share/common-licenses/GPL-2'.
```

Anatomy of Debian Source Package

- `debian/control`
  - Source package metadata
  - Binary packages metadata

```
Source: package
Maintainer: Joe Black <joe@abc.com>
Section: misc
Priority: optional
Build-Depends: debhelper (>= 8.1.3)
Standards-Version: 3.9.2

Package: package
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: An example package
 An example package demonstrating how to create a Debian
 package.
```

Anatomy of Debian Source Package

- `debian/source/format`
  - Specifies source format version
    - `3.0 (quilt)`
    - `3.0 (native)`
    - `1.0`
- `debian/compat`
  - debhelper compatibility level
  - New packages should use 7
  - See `debhelper(7)` man page for more info

Anatomy of Debian Source Package

- `debian/rules`
  - `make` rules with specific set of targets:
    - `clean` ≈ `make clean`
    - `build` ≈ `make`
    - `binary` ≈ `make install` & pack
    - `binary-arch` – only build arch-dependent binaries
    - `binary-indep` – only build arch-independent binaries
  - However, with debhelper 7:

```
#!/usr/bin/make -f
%:
        dh $@
```

  - You can also override the default `dh` rules

- debhelper
  - Commands to help doing tasks in package building process, e.g.
    - `dh_prep` – Clean & prepare the build directory
    - `dh_install` – Install files to target directory
    - `dh_compress` – Compress files where they should
    - `dh_fixperms` – Fix file permissions
    - `dh_shlibdeps` – Calculate shared library dependencies
    - `dh_mkshlibs` – Create library info for shared lib dependency calculation
    - etc.
  - To be called from `debian/rules`
  - Actually command-line commands, *with man pages*

- debhelper
  - In debhelper 7+:
    - `dh_*` commands are shortened to `dh *`
    - `dh auto_*` commands added to call usual series of `dh *` commands
    - `dh` *`target`* for `debian/rules` targets to call appropriate `dh auto_*` commands
    - Thus, the rules become:

```
#!/usr/bin/make -f
%:
        dh $@
```

    - The defaults can be overridden
    - Add-ons for common overrides (e.g. `dh-autoreconf`)

- CDBS (Common Debian Build System)
  - Abstract build rules based on Makefile inheritance
  - Used for packages of common classes
    - Perl
    - GNOME
    - KDE
    - cmake
    - waf
    - etc.
  - Common rules are maintained at a single place

Patch Management Systems

- dpatch
  - LIFO patch management
  - `dpatch list-all` – List all patches
  - `dpatch patch` *patch-name* – Apply patches until *patch-name*
  - `dpatch patch-all` – Apply all patches
  - `dpatch unpatch` *patch-name* – De-apply patches until *patch-name*
  - `dpatch unpatch-all` – De-apply all patches
  - Editing patch
    - `dpatch-edit-patch patch` *patch-name* – Edit patch *patch-name*, add new if not exists
    - Copies the whole source tree to a temporary dir
    - Creates a shell for user to edit files
    - When exit from shell, diff the source and update the patch
  - Convenient for editing & testing
  - Not good for large source tree
  - Tedious command line (patch name required everywhere)
  - Obsolete soon

Patch Management Systems

- quilt
  - LIFO patch management
  - `quilt series` – List all patches
  - `quilt push` – Apply next patch in series
  - `quilt push -a` – Apply all patches
  - `quilt pop` – De-apply current patch
  - `quilt pop -a` – De-apply all patches
  - Editing patch
    - `quilt new` *patch-name* – Create a new patch *patch-name*
    - `quilt add` *file* – Add a file to keep track of changes
    - Current copy copied as the base
    - User edits the file
    - `quilt refresh` – Re-diff the files and update the patch
  - Add before edit
  - Light weight, copy on demand, good for source of any size
  - Simple command line
  - Adopted as new standard

Good Patching Practices

- Do not repack upstream tarball (except for licensing issues)
- Always use a patch system (quilt/dpatch)
- Make sure there is no excessive changes after build

Logging Changes

- Debian package version is determined by `debian/changelog`
- Don't forget to log changes before building modified version
- Command: `dch`/`debchange`
  - `dch` – Log change without version change
  - `dch -i` – Increment Debian version
  - `dch --nmu` – Increment Debian version for NMU
  - `dch -v` *`newver`* – Force new version (e.g. new upstream version)

Package Versioning

Format:

[Epoch :] UpstreamVersion [-DebianVersion]

- Epoch (optional) – small number for overriding old versioning scheme. **Use sparingly!**
- UpstreamVersion – version released by upstream
- DebianVersion – version of the Debian package based on the same upstream version
  - Maintainer's version – integer, starts at 1, incremented by one
  - Non-maintainer's version – decimal point incremented from the latest maintainer's version, starts at .1, incremented by .1
- Debian-native packages:
  - DebianVersion is omitted
  - `debian/source/format` : "`3.0 (native)`"

Packaging New Software

- Tool: dh-make
  - Install:

```
sudo apt-get install dh-make
```

  - Packaging:
    - Prepare upstream tarball:

```
mv package-version.tar.gz \
package_version.orig.tar.gz
```

    - Extract source:

```
tar xzf package_version.orig.tar.gz
```

    - Invoke `dh_make`:

```
cd package-version
dh_make
```

    - Select package type (single binary, indep binary, multiple binary, library, kernel module, kernel patch)
    - Confirm package info
    - Edit template `debian/*` files

debian/control

- Standards-Version:
  - Debian Policy version the package claims to conform to
  - Changes between Policy versions: `/usr/share/doc/debian-policy/upgrading-checklist.txt.gz`
- Section:
  - admin, cli-mono, comm, database, devel, debug, doc, editors, electronics, embedded, fonts, games, gnome, graphics, gnu-r, gnustep, hamradio, haskell, httpd, interpreters, java, kde, kernel, libs, libdevel, lisp, localization, mail, math, misc, net, news, ocaml, oldlibs, otherosfs, perl, php, python, ruby, science, shells, sound, tex, text, utils, vcs, video, web, x11, xfce, zope
- Priority:
  - required (for dpkg to function)
  - important (basic Unix)
  - standard (reasonable default text-mode installation)
  - optional (larger systems like X Window, TEX)
  - extra (other than above)

- Dependencies for source packages
  - Build-Depends:
    - needed for building
  - Build-Depends-Indep:
    - needed for building arch-indep part

- Dependencies for binary packages
  - Depends:
    - required
    - package configured only if the dependencies are configured
  - Recommends:
    - not required, but usally installed together
  - Suggests:
    - not installed together is usual, but can enhance
  - Enhances:
    - reverse Suggests:
  - Pre-Depends:
    - required
    - package *unpacked* only if the dependencies are configured
    - usually used for commands used in preinst

- Breaking & conflicting binary packages
  - Breaks:
    - package configured only if the broken is de-configured
    - still allow simultaneous unpack
    - usually used with << to enforce upgrade on the broken package first
  - Conflicts:
    - no simultaneous unpack is allowed
  - Replaces:
    - take over the files from the other package

- Package short description
  - One-line summary
- Package long description
  - Multi-paragraph detailed description
  - Tells what the package provides
  - Distinction from other similar packages (if any)
  - For each sub-package, describe which part it provides

Maintainer Scripts

- Shell scripts to be run on different installation stages:
  - preinst
  - postinst
  - prerm
  - postrm
- With different command-line args for different operations (install, configure, upgrade, remove, purge, abort, etc.)
- See Debian Policy Chapter 6 for more details

Triggers

- dpkg facility to collect & pass events caused by a set of packages to another during installation
- Currently supported triggers:
  - Explicit triggers – by calling `dpkg-trigger`
  - File triggers – fired on file changes
- File trigger: Interested package:
  - `debian/triggers`:

    `interest` *`/path/to/directory/or/file`*
  - "`postinst triggered`" called when triggered
- More details:
  `/usr/share/doc/dpkg-dev/triggers.txt.gz`

debian/watch

- Optional file for tracking new upstream release

```
version=3
# Full-site-with-pattern [Version [Action]]
ftp://upstream.host/path/package-([\d+\.]+|\d+)\.tar\.gz
debian uupdate
```

- Used by `http://dehs.alioth.debian.org/` and other Debian QA tools
- "`uscan`" command
  - Checks upstream files compared to version in second field ("`debian`" or if omitted means latest in `debian/changelog`)
  - Newer upstream found → download
  - Run command in third field if specified (usually `uupdate`, or omitted for no command)

QA Tools

- `lintian`
  - Checks for common mistakes in packages (source & binary)
  - Official packages should be lintian-clean
  - New checks keep added → new issues to solve
  - `lintian.debian.net`
  - "`lintian`" command:
    - Argument can be `.changes`, `.dsc` or `.deb` file
    - "`-i`" option turns on more checks (info)
    - "`-I`" option to show detailed explanation for issues

QA Tools

- `piuparts` (Package Installation, UPgrade And Removal Testing Suite)
  - Try installing, upgrading, removing package in a clean chroot
  - Check if the operations go well without error
  - Check if any file is left after removal (e.g. generated files)
  - Use base system from tarball, same as pbuilder
  - Example:
    - `piuparts foo_1.0-2_amd64.deb`
    - `piuparts foo_1.0-2_amd64.changes`

Section 5

Delivering Packages

Officially

- Check WNPP (Work-Needing and Prospective Packages) first
- `http://www.debian.org/devel/wnpp/`
- WNPP bug types
  - ITP (Intent To Package)
  - O (Orphaned)
  - RFA (Request For Adoption)
  - RFH (Request For Help)
  - RFP (Request For Package)
- Taking responsibility on a WNPP
  - ITP $\rightarrow$ the submitter is already responsible for it
  - O $\rightarrow$ ITA (Intent To Adopt)
  - RFA $\rightarrow$ ITA
  - RFH $\rightarrow$ to be closed by submitter
  - RFP $\rightarrow$ ITP

Officially

- Closing WNPP bug
  → upload the package to Debian with `debian/changelog` entry:
  - For ITP bug:

```
* Initial release (Closes: #XXXXXX)
```

  - For ITA bug:

```
* New maintainer (Closes: #XXXXXX)
```

- No upload right? → Request for sponsorship
  - A DD can *sponsor* your upload by doing it for you using his/her trusted digital signature
  - He/she shares the responsibility with you
    → You will be asked to correct things
  - Upload place: `http://mentors.debian.net`
  - RFS (Request For Sponsorship) to `debian-mentors@lists.debian.org`
  - IRC channel `#debian-mentors` @ OFTC as supplement

Custom Repository

reprepro

- What it does
  - Adds/removes/updates package files in repository
  - Generates apt index files
  - Input: `.changes` file
- Setting up a repository
  - `cd` *`repo`*
  - `mkdir conf`
  - `vi conf/distributions`

```
Codename: mysid
Components: main contrib non-free
Architectures: i386 amd64 source
```

Custom Repository

- More detailed configuration

```
Codename: mysid
Suite: unstable
Components: main contrib non-free
UDebComponents: main contrib non-free
Architectures: i386 amd64 source
Origin: mycompany
Version: 1.0
Description: mycompany repository
AlsoAcceptFor: unstable
```

- Multiple distributions are separated by empty lines
- Create suite → codename symlinks

```
reprepro -b /your/base/dir createsymlinks
```

Custom Repository

- Include new packages

```
reprepro -b /your/base/dir include codename .changes-file
```

- Remove package

```
reprepro -b /your/base/dir removesrc codename source-name
```

- Migrate package across distributions

```
reprepro -b /your/base/dir copysrc dest-codename
src-codename source-name
```

- **Note:** *Never* manage files manually. *Always* do it via reprepro commands. (So database is updated.)

Custom Repository

- Upload handling
  - `vi conf/incoming`

```
Name: mycompanyincoming
IncomingDir: /home/ftp/mycompany/incoming
TempDir: /home/ftp/mycompany/tmp
Allow: unstable>mysid
Default: mysid
Cleanup: on_deny on_error
```

  - `reprepro` command for incoming handling

```
reprepro -b /your/base/dir processincoming rule-name
.changes-file
```

Custom Repository

- Automatic incoming handling
  - `sudo apt-get install inoticoming`
  - `sudo vi /etc/default/inoticoming`

```
USER=repo
LOGFILE=/var/log/incoming.log
INITIALSEARCH=0
DIR=/home/ftp/mycompany/incoming

# actions
BASEDIR=/home/ftp/mycompany
RULENAME=mycompanyincoming
ACTIONS="--suffix .changes --stderr-to-log \
reprepro -s -b $BASEDIR --waitforlock 1000 \
processincoming $RULENAME {} ;"
```

  - Any new file with `.changes` suffix under incoming dir will trigger the inoticoming action

Custom Repository

- Trusted uploaders
  - To prevent intruder's uploads, verify GPG signatures before processing changes
  - `Uploaders:` fields in `conf/distributions` specify trusted uploader key list file for each distribution

```
Uploaders: uploaders
```

  - `SignWith:` fields in `conf/distributions` for signing the `Release` file

```
SignWith: 0xXXXXXXXX
```

  - `vi conf/uploaders`

```
# Joe Black
allow * by key 0xXXXXXXXX

# Bill Parrish
allow * by key 0xYYYYYYYY
```

  - The GPG keys must be available to inoticoming user's public keyring

Custom Repository

- More info:
  - `/usr/share/doc/reprepro/manual.html`
  - `reprepro(1)` man page

Live CD

- Live CD tools in Debian
  - live-build – low-level tool for live CD creation
  - live-magic – GUI frontend for live-build
- Live CD types
  - iso – for live CD/DVD only
  - usb – for live USB thumdrive only
  - iso-hybrid – single image for both CD/DVD and USB
- Live features
  - Live system without installation
  - Integrated installer by copying live image to harddisk
  - Rescue disc

- Configuration

```
lb config -b iso-hybrid -d wheezy -a i386 -k 686 \
  -m http://192.168.1.1:9999/debian \
  --mirror-binary http://ftp.th.debian.org/debian \
  --archive-areas "main contrib non-free" \
  --apt-options "-y --auto-remove" \
  --bootloader syslinux \
  --syslinux-theme debian-squeeze \
  --debian-installer live \
  --debian-installer-distribution daily \
  --debian-installer-gui false \
  --package-lists "standard" --tasks "laptop" \
  --hostname mysid --username live
```

- Build

```
sudo lb build
```

- Customization
  - `config/chroot_local-packagelists/*.list`
    List of packages from mirror to install in live system
  - `config/chroot_local-packages/*.deb`
    Local packages to install in addition
  - `config/chroot_local-includes/*`
    Files to add to the live system root FS (e.g. config files)
  - `config/chroot_local-includes/etc/live/config.conf`
    The live system configuration
  - `config/binary/isolinux/*`
    Files to customize the CD boot loader
  - `config/binary_debian-installer/*`
    Files to customize the installer (e.g. preseed.cfg)
- More info
  - `http://live.debian.net/manual/`