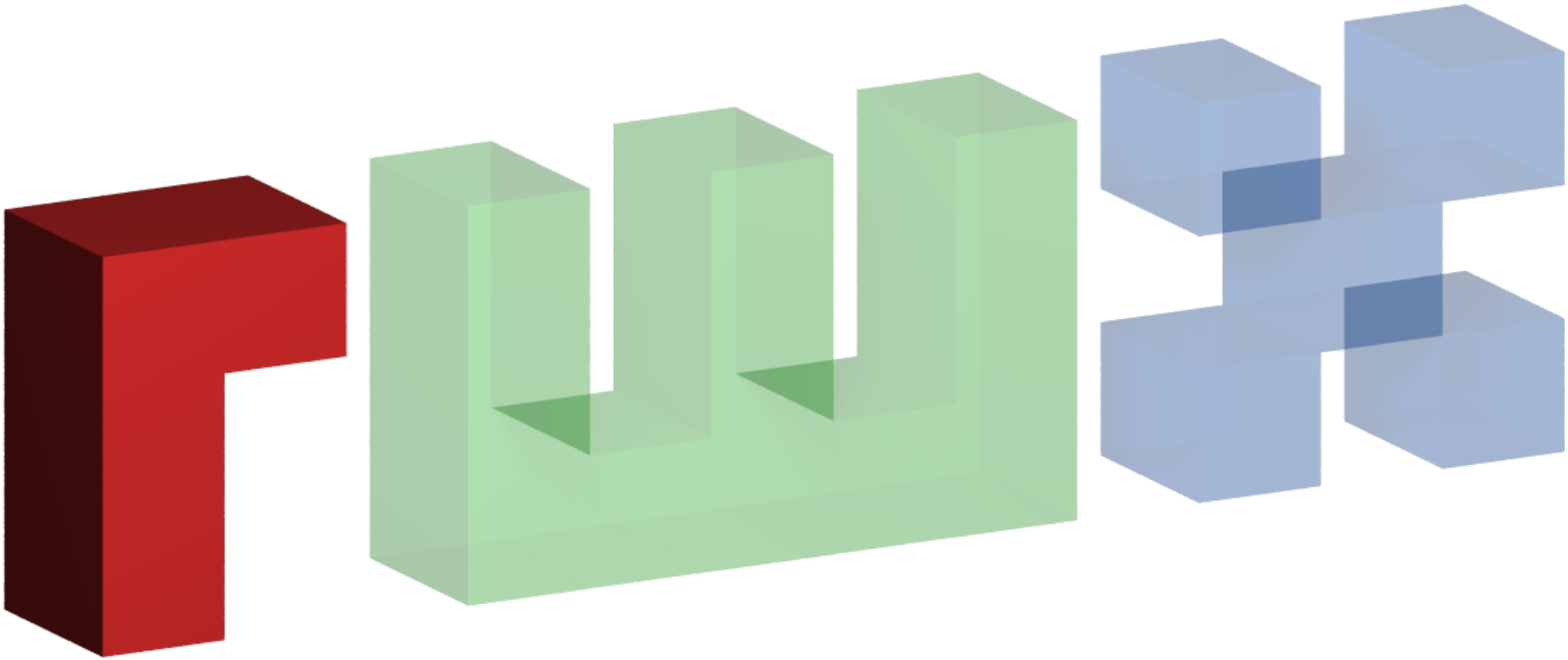
User- and distro-friendly packaging

Recommendations to upstream tarball providers

Jan Engelhardt <jengelh@inai.de>

Presented at the openSUSE Conference 2011

2011-Sep-11–14



This talk is R-rated :)

Table of Contents

Abstract

This just for the non-live audience.

Even after the release of a software tarball, there remains work to be done. Not all end-users run software from tarballs, but often rely on a form of infrastructure to (re)provide the software in a fashion of their liking — e. g. a Linux distro shipping compiled binaries —, and this transition is not completely automatable. This talk wants to give direction on how upstream software providers can reduce the workload for packagers in the downstream direction, and thus increase chances that your software will be available to more end-users.

Nomenclature

Software Propagation Line

- upstream developers: LOC crafting
- release manager: package and release source code archive (usually a "tarball"), and keeper of the GPG key
- distro contributor: takes tarball and produces a RPM package
- end-user: installation of tarball or RPM package

Target audience

- upstream developers: LOC crafting
- release manager: package and release source code archive (usually a "tarball"), and keeper of the GPG key
- distro contributor: takes tarball and produces a RPM package
- end-user: installation of tarball or RPM package

Why You Should Care (1/3)

- You retain the freedom to design your tarball releases
- presenting guidelines — not binding, but recommended nevertheless

Why You Should Care (1/3)

- You retain the freedom to design your tarball releases
- presenting guidelines — not binding, but recommended nevertheless
- it's about the users
  - direct: end-users
  - indirect: administrators, distro contributors, help desk/support line

Why You Should Care (2/3)

It's all about the users...

- faster for them to apply known paradigms (e. g. "`./configure && make install`")
- users don't want to RTFM
- provided docs even exist

Why You Should Care (2/3)

It's all about the users...

- faster for them to apply known paradigms (e. g. "`./configure && make install`")
- users don't want to RTFM
- provided docs even exist

Why You Should Care (2/3)

It's all about the users...

- faster for them to apply known paradigms (e. g. "`./configure && make install`")
- users don't want to RTFM
- provided docs even exist

When things go wrong

- negative reviews much have more weight

Why You Should Care (2/3)

It's all about the users...

- faster for them to apply known paradigms (e. g. "`./configure && make install`")
- users don't want to RTFM
- provided docs even exist

When things go wrong

- negative reviews much have more weight
- software ratings (incl. word of mouth) playing a role
- $\Rightarrow$ upset as few users as possible

Why You Should Care (3/3)

And distros...

- want to contain rank growth (de:Wildwuchs)
- remember Factory has some 4600+ packages
- fewer deviations $\rightarrow$ fewer code, fewer chances for bugs

Why You Should Care (3/3)

And distros...

- want to contain rank growth (de:Wildwuchs)
- remember Factory has some 4600+ packages
- fewer deviations $\rightarrow$ fewer code, fewer chances for bugs

And you want users, don't you?

- distro might just silently "fix" it to provide a better package experience

Versioning Fundamentals

- sequenced version numbers ($3.1 < 3.1.4 < 3.14$)
  - tools like rpm etc. use this interpretation
- other versioning schemes need conversion e. g. the decimal point scheme ($3.1 < 3.14 < 3.2$)
  - $3.141 \rightarrow 3.1.4.1$
  - thankfully not widespread

Versioning Fundamentals

- sequenced version numbers ($3.1 < 3.1.4 < 3.14$)
  - tools like rpm etc. use this interpretation
- other versioning schemes need conversion e. g. the decimal point scheme ($3.1 < 3.14 < 3.2$)
  - $3.141 \rightarrow 3.1.4.1$
  - thankfully not widespread (yes, I am ware of TEX)

Source Archive Basics (1/4)

- name of the archive file: should contain name and version (or usually just name for tarred-up snapshots)
- archive should extract into a new directory, not `$PWD`

```
$ tar -xvvf foo-1.0.tar.xz
-rw-r--r-- root/root 283 2011-09-12 configure.ac
```

Install into a new directory

```
$ tar -xvvf foo-1.0.tar.xz
-rw-r--r-- root/root 283 2011-09-12 foo-1.0/configure.ac
```

Source Archive Basics (2/4)

- top-level extracted directory's version (if any) should match archive version

```
$ tar -xvvf gap4r4p12.tar.xz
-rw-r--r-- root/root 283 2011-09-12 gap4r4/configure
```

Top-level directory's version (if any) should match archive version

```
$ tar -xvvf foo-1.0.5.tar.xz
-rw-r--r-- root/root 283 2011-09-12 foo-1.0.5/configure
```

Source Archive Basics (3/4)

- omit redundant components

Just one build per version so far; makes build number redundant

VirtualBox-**4.1**-4.1.2_**73507**_openSUSE114-1.x86_64.rpm
VirtualBox-**4.1**-4.1.0_**73009**_openSUSE114-1.x86_64.rpm
VirtualBox-**4.0**-4.0.12_**72916**_openSUSE114-1.x86_64.rpm

- version provided in `%name` also redundant, since vbox-4.1 cannot be installed alongside vbox-4.0

Source Archive Basics (4/4)

- preferably do not mix separators;
  avoid “releases”, only do “versions” (rpm/deb lingo):
  `ebtables-v2.0.10-2.tar.gz` → `ebtables-2.0.10.2.tar.gz` ✓
- dots preferred as a separator within distros
  `gap4r4p12.tar.gz` → `gap-4.4.12.tar.gz`

Archive Formats (1/2)

- `.tar` is the container of choice,
  - with `.gz` (oldskool) or `.xz` (modern)
- also circulating: `.zip` (archaic), `.7z`

Archive Formats (2/2)

- recompressed and/or repackaged at times to different format
- reduction of BRPM/SRPM sizes

Archive Formats (2/2)

- recompressed and/or repackaged at times to different format
- reduction of BRPM/SRPM sizes
- object files normally should not be in the archive
- much less in an SCM

To Split or Not To Split?

[ : This frame has +w capabilities.]

- big packages introduce serialization/limit scaleout
- split generic libraries from big projects:
  - samba → libtalloc, libtdb, libtevent
  - util-linux → libblkid, libmount, libuuid
- documentation often likes to be put into the main package
- success stories?
  - *-data packages, mostly in the “games” project

Source Availability

Do provide the source in a SCM:

- to grab the HEAD and make patches against that instead
- rather important if tarball releases reflect a state too old to receive patches,
  - classical long-winded release cycles
  - and where stable and devel have diverged quite a bit
  - gap 4.4 vs. 4.5
  - Linux 2.4 vs. 2.5
- can see when and how my patch was applied
- can package the SCM HEAD and drop my local patch

Source-level Build System

Claim: plain Makefiles are the most buggy ones. Examples:

- wrongful population of `CFLAGS`, which is a user-overridable variable, with essential flags
- failure to properly implement `DESTDIR` for ‘`make install`‘
- parallel make rendered non-functional

Source-level Build System

Claim: plain Makefiles are the most buggy ones. Examples:

- wrongful population of `CFLAGS`, which is a user-overridable variable, with essential flags
- failure to properly implement `DESTDIR` for '`make install`'
- parallel make rendered non-functional

Use some common (source-level) build automation system.

- C, C++: automake, cmake, scons, ...
- Python: setup.py
- Perl: Makefile.PL

Source-level Build Systems (2/2)

$BUILD_SYS

- no need to figure out names of variables in Makefiles
- use BS for consistency
  - autoconf: options commonly start with `--enable-` or `--with-`
  - ...
- for simplicity, bug surface reduction
  - automake: installation taken care of automatically

Compilation (C, C++) (1/2)

- enable warnings, and address them

`-Wall` is not enough; useful `CFLAGS`:

```
-D_FORTIFY_SOURCE=2 -Wall -Waggregate-return
-Wmissing-declarations -Wmissing-prototypes
-Wredundant-decls -Wshadow -Wstrict-prototypes -Wformat=2
```

- compiler may warn, but incorrectly, or only half of the time
- fixing itself requires in-depth knowledge of the used language

Compilation (2/2)

```
printf("%d\n", sizeof(int));
```

Emitted Warning (and only on 64-bit!)

test.c:4:2: warning: format “%d” expects argument of type “int”, but argument 2 has type “long unsigned int” [-Wformat]

Should we tr...

%ld?

Compilation (2/2)

```
printf("%d\n", sizeof(int));
```

Emitted Warning (and only on 64-bit!)

test.c:4:2: warning: format “%d” expects argument of type “int”, but argument 2 has type “long unsigned int” [-Wformat]

Bzzzt

```
%ld
```

```
printf("%zu\n", sizeof(int));
/* or in the absence of C99:  */
printf("%lu\n", (long)sizeof(int));
```

Library Versioning (1/2)

Library Versioning (1/2)

libqt4-**4.7.1**: `libQtCore.so.4.7.1`

- so-version equaling the package version is a sign of a possible error
- other platforms do not necessarily use the same three-component style for SO versions
- meaning of SO versions is nevertheless consistent across platforms

Library Versioning (1/2)

libqt4-**4.7.1**: `libQtCore.so.4.7.1`

- so-version equaling the package version is a sign of a possible error
- other platforms do not necessarily use the same three-component style for SO versions
- meaning of SO versions is nevertheless consistent across platforms

How SO versions work

Qt version	linux SO ver	means API range	BSD SO ver
4.7.x	`4.7.x`		

Library Versioning (1/2)

libqt4-**4.7.1**: `libQtCore.so.4.7.1`

- so-version equaling the package version is a sign of a possible error
- other platforms do not necessarily use the same three-component style for SO versions
- meaning of SO versions is nevertheless consistent across platforms

How SO versions work

Qt version	linux SO ver	means API range	BSD SO ver
4.7.x	`4.7.x`	4–11	`.so.11`
4.8.x	`4.8.x`		

Library Versioning (1/2)

libqt4-**4.7.1**: `libQtCore.so.4.7.1`

- so-version equaling the package version is a sign of a possible error
- other platforms do not necessarily use the same three-component style for SO versions
- meaning of SO versions is nevertheless consistent across platforms

How SO versions work

Qt version	linux SO ver	means API range	BSD SO ver
4.7.x	`4.7.x`	4–11	`.so.11`
4.8.x	`4.8.x`	4–12	`.so.12`
5.6.x	`5.6.x`		

Library Versioning (1/2)

libqt4-**4.7.1**: `libQtCore.so.4.7.1`

- so-version equaling the package version is a sign of a possible error
- other platforms do not necessarily use the same three-component style for SO versions
- meaning of SO versions is nevertheless consistent across platforms

How SO versions work

Qt version	linux SO ver	means API range	BSD SO ver
4.7.x	`4.7.x`	4–11	`.so.11`
4.8.x	`4.8.x`	4–12	`.so.12`
5.6.x	`5.6.x`	5–11↯	`.so.11`↯

Library Versioning (2/2)

Actually *do* SO versioning

libx264 `libx264.so.115`

Library Versioning (2/2)

Actually *do* SO versioning

libx264 `libx264.so.115`

When not caring about SO ver.: change basename to ensure uniqueness

binutils-2.21 `libbfd-2.21.so`

hunspell-1.2.12 `hunspell-1.2.so.0.0.0`

Library Versioning (2/2)

Actually *do* SO versioning

libx264 `libx264.so.115`

When not caring about SO ver.: change basename to ensure uniqueness

binutils-2.21 `libbfd-2.21.so`

hunspell-1.2.12 `hunspell-1.2.so.0.0.0`

...or combine both

libpng-1.2 `libpng12.so.0.46.0`

libpng-1.4 `libpng14.so.14.4.0`

glib-2.28.0 `libglib-2.0.so.0.2800.0` [a]

[a] arguably, `libglib-2` as a basename would have been enough

e^3

e^3

- Extract: look for distros' build scripts, and build logs of your package
  - `build.opensuse.org`
  - `packages.debian.org`

e^3

- Extract: look for distros' build scripts, and build logs of your package
  - `build.opensuse.org`
  - `packages.debian.org`
- Examine: see how they build it, see what workarounds they made
  - CDBS has many architectures $\rightarrow$ new compile errors/warnings
  - OBS: rpmlint and BRP results

e^3

- Extract: look for distros' build scripts, and build logs of your package
  - `build.opensuse.org`
  - `packages.debian.org`
- Examine: see how they build it, see what workarounds they made
  - CDBS has many architectures $\rightarrow$ new compile errors/warnings
  - OBS: rpmlint and BRP results
- Enhance: merge useful parts preemptively, do not wait for distro to submit

- stick to conventions, make it easy for users
- if you get patches that turn your Makefile into automake (or something else), do consider it
- do not be alarmed if someone runs `spec-beautifier` (or `spec-cleaner` / `obs-service-format_spec_file`)

Ask if assistance needed:

User- and distro-friendly packaging

Recommendations to upstream tarball providers

Jan Engelhardt <jengelh@inai.de>

Presented at the openSUSE Conference 2011

2011-Sep-11–14