Fedora Packaging

Deepak Bhole
Andrew Overholt

2008-10-07
Outline

● What is packaging and why is it important?
● How are packages built?
● How does Fedora maintain its packages?
What are packages?

- Binary Linux distributions are made up of modular components called “packages”
- Package = archive of files + metadata
Why use packages?

- Standardize deployments
- Query installation information
- Standards compliance
- Security and auditibility
- Dependency management
What is packaging?

- Process of creating packages
What is packaging?

- Process of creating RPMs
An Example

- **helloworld-1.0.tar.gz**
  - Upstream source release
  - `tar zxf helloworld-1.0.tar.gz`
  - `./configure; make; make install`

- **helloworld.spec**
  - Like a recipe
  - Includes metadata like requirements
  - Scripted build, “install”
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog
Anatomy of a `.spec`

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog
- Name/Version/Group/License
# Preamble

**Name:** helloworld  
**Version:** 1.0  
**Release:** 1{%?dist}  
**Summary:** Hello, world!  

**Group:** Applications  
**License:** GPLv2+  
**URL:** http://foo.ca/helloworld  
**Source0:** %{name}-%{version}.tar.gz  
**Patch0:** %{name}-%{version}-typo.patch  
**BuildRoot:**  
%{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)  

%description  
Hello, world is amazing.
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Name/Version/Group/License
- Release (tracks build changes)
Preamble

Name: helloworld
Version: 1.0
Release: 1%{?dist}
Summary: Hello, world!

Group: Applications
License: GPLv2+
URL: http://foo.ca/helloworld
Source0: %{name}-%%{version}.tar.gz
Patch0: %{name}-%%{version}-typo.patch
BuildRoot: %{_tmppath}/%{name}-%%{version}-%%{release}-root-%(%{__id_u} -n)

description
Hello, world is amazing.
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Name/Version/Group/License
- Release (tracks build changes)
- Sources/Patches
Preamble

Name:           helloworld
Version:        1.0
Release:        1%{?dist}
Summary:        Hello, world!

Group:          Applications
License:        GPLv2+
URL:            http://foo.ca/helloworld
Source0:        %{name}- %{version}.tar.gz
Patch0:         %{name}- %{version}-typo.patch
BuildRoot:
    %{_tmppath}/%{name}- %{version}- %{release}- root- %(%{__id_u} -n)

Requires:       pango
BuildRequires:  pango-devel
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Name/Version/Group/License
- Release (tracks build changes)
- Sources/Patches
- Requirements
  - Both build & install
Preamble

Name: helloworld
Version: 1.0
Release: 1%{?dist}
Summary: Hello, world!

Group: Applications
License: GPLv2+
URL: http://foo.ca/helloworld
Source0: %{name}-%{version}.tar.gz
Patch0: %{name}-%{version}-typo.patch
BuildRoot: %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)

Requires: pango
BuildRequires: pango-devel
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Name/Version/Group/License
- Release (tracks build changes)
- Sources/Patches
- Requirements
  - Both build & install
- Summary/Description
Preamble

Name: helloworld
Version: 1.0
Release: 1%{?dist}
Summary: Hello, world!

Group: Applications
License: GPLv2+
URL: http://foo.ca/helloworld
Source0: %{name}-%{version}.tar.gz
Patch0: %{name}-%{version}-typo.patch
BuildRoot: %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)

description
Hello, world is amazing.
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Name/Version/Group/License
- Release (tracks build changes)
- Sources/Patches
- Requirements
  - Both build & install
- Summary/Description
- Custom macro definitions
Custom macros

%ifarch %{ix86}
%define eclipse_arch x86
%else
%define eclipse_arch %{_arch}

Summary: An open, extensible IDE
Name: eclipse
...
...
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Sub-package definitions
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Sub-package definitions

```spec
%package devel

Summary: API documentation for helloworld

Group: Documentation

Requires: %{name} = \\
  %{version}- %{release}

%description devel

This package contains the API documentation for helloworld.
```
Anatomy of a .spec

- Preamble
- **Prep**
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Sources unpacked here
Anatomy of a .spec

- Preamble
- **Prep**
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Sources unpacked here
- **Patches applied**
Anatomy of a .spec

- Preamble
- **Prep**
- Build
- Install
- Clean
- **Scriptlets**
- Files
- Changelog

- Sources unpacked here
- Patches applied
- **Example:**
  
  ```
  %prep
  %setup -q
  %patch0 -p0
  ```
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- %configure macro has good defaults
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- \%configure macro has good defaults
- **Example of a \%build section:**
  \%build
  \%configure
  make \%{\_smp_mflags}
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Creates buildroot
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Creates buildroot
- Lays out filesystem structure
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Creates buildroot
- Lays out filesystem structure
- Puts built files into buildroot
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

**Example:**

```
%install
rm -rf %{buildroot}
install -dm 755 \
   %{buildroot}/%{_bindir}
%{buildroot}/%{__bindir}
cp -p mybinary \
   %{buildroot}/%__bindir
```

**Example:**

```
%install
rm -rf %{buildroot}
make install \ 
   DESTDIR=$RPM_BUILD_ROOT
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- **Clean out the buildroot**
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Clean out the buildroot
  - Example:
  - `%clean`
  - `rm -rf %{buildroot}`
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- **Scriptlets**
- Files
- Changelog

- **Scripts that get run at various stages of installation**
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- **Scriptlets**
- Files
- Changelog

- Scripts that get run at various stages of installation

- **Four commonly-used sections:**
  - %pre
  - %preun
  - %post
  - %postun
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

**Warning!** Be sure to read the `triggers_order` section of

/usr/share/doc/rpm-*/triggers
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

Example:

```
%post
if [ -x %{_bindir}/rebuild-gcj-db ]
    then
        %{_bindir}/rebuild-gcj-db
    fi
```
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- List of package contents
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- List of package contents
- If it is not in %files, it is not in the package!
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- List of package contents
- If it is not in %files, it is not in the package!
- RPM will complain about un-packaged and multiply-listed files – check warnings
Anatomy of a `.spec`

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

**Example:**

```
%files
%defattr(-,root,root,-)
%doc COPYING README
%{_bindir}/%{name}
```
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

**Example:**

```
%files
%defattr(-,root,root,-)
%doc COPYING README
{%_bindir%/}%{name}
```

**Sub-package:**

```
%files devel
%defattr(-,root,root,-)
%doc APIDOCs
```
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Used to track package changes
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Used to track package changes
- Not intended to replace source code ChangeLog
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Used to track package changes
- Not intended to replace source code
  ChangeLog
- Provides explanation, audit trail
Anatomy of a .spec

- Preamble
- Prep
- Build
- Install
- Clean
- Scriptlets
- Files
- Changelog

- Used to track package changes
- Not intended to replace source code
- ChangeLog
- Provides explanation, audit trail
- Update on every change
Set up for building

- Make an rpmbuild tree in your home directory:
  
rpmdev-setuptree (in rpmdevtools package)
- The above will also create ~/.rpmmacros:
  
  %_topdir %(echo $HOME)/rpmbuild
Build

- `cp *.gz *.patch ~/rpmbuild/SOURCES`
- `cp *.spec ~/rpmbuild/SPECS`
- `rpmbuild -ba \ 
  ~/rpmbuild/SPECS/helloworld.spec`
Results

- `ls ~/rpmbuild/RPMS/$arch`
  helloworld-1.0-1.fc9.x86_64.rpm
  helloworld-debuginfo-1.0-1.fc9.x86_64.rpm
  helloworld-devel-1.0-1.fc9.x86_64.rpm

- `ls ~/rpmbuild/SRPMS`
  helloworld-1.0-1.fc9.src.rpm
Catching common mistakes

- **rpmlint**

  $ rpmlint helloworld-1.0-1.fc9.x86_64.rpm
  helloworld.x86_64: W: non-standard-group Applications
  1 packages and 0 specfiles checked; 0 errors, 1 warnings.

  $ rpmlint -I non-standard-group Applications
  non-standard-group:
  The value of the Group tag in the package is not valid.
  Valid groups are:
  ...

What is packaging?

- Process of creating **RPMs for Fedora**
Fedora packaging

• Goal:
  – RPM of upstream “stuff”, available for installation via the internet or DVD/CD
  – ex.

  yum install helloworld
Fedora packaging infrastructure

- CVS
Fedora CVS

$ ls -1
common
CVS
devel
F-7
F-8
F-9
import.log
Makefile
pkg.acl
Fedora packaging infrastructure

- CVS
- customized Makefiles
Fedora packaging infrastructure

- CVS
- customized Makefiles
- Koji (build system)
Fedora packaging infrastructure

- CVS
- customized Makefiles
- Koji (build system)
- Bodhi (update system)
Fedora packaging

- Process:
  - Make
  - Review
  - Build
  - Distribute

Fedora packaging

- Process:
  - Make
  - Review
  - Build
  - Distribute

- Get source
Fedora packaging

• Process:
  – Make
  – Review
  – Build
  – Distribute

• Get source
• Make .spec
Fedora packaging

• Process:
  – Make
  – Review
  – Build
  – Distribute

• Get source
• Make .spec
• Build and test locally
Fedora packaging

- Process:
  - Make
  - Review
  - Build
  - Distribute

- Get source
- Make .spec
- Build and test locally
- **Verify that it meets guidelines**

http://fedoraproject.org/wiki/Packaging/Guidelines

http://fedoraproject.org/wiki/Packaging/NamingGuidelines
Fedora packaging

• Process:
  – Make
  – Review
  – Build
  – Distribute

• Reviews ensure consistency and maintainability

http://fedoraproject.org/wiki/Packaging/ReviewGuidelines
Fedora packaging

• Process:
  – Make
  – Review
  – Build
  – Distribute

• Submit package for review
Fedora packaging

- Process:
  - Make
  - Review
  - Build
  - Distribute

- Submit package for review

- Package is reviewed
Fedora packaging

- Process:
  - Make
  - Review
  - Build
  - Distribute

- Submit package for review
- Package is reviewed
- Upon acceptance, CVS module created, package uploaded
Fedora packaging

- Process:
  - Make
  - Review
  - Build
  - Distribute

- make tag
Fedora packaging

● Process:
  – Make
  – Review
  – Build
  – Distribute

● make tag
● make local
Fedora packaging

• Process:
  – Make
  – Review
  – Build
  – Distribute

• make tag
• make local
• make build
Fedora packaging

• Process:
  – Make
  – Review
  – Build
  – Distribute

• make tag
• make local
• make build
• make help
Tips

- rpmdevtools
  - rpmdev-vercmp
- gendiff
- Local mock build
- **Read the koji logs**
  - build.log, root.log, etc.
- make scratch-build
- #fedora-devel
Fedora packaging

• Process:
  – Make
  – Review
  – Build
  – Distribute

• magic
Upstream

- Upstream, upstream, upstream
Links

- Fedora Packaging Guidelines: http://fedoraproject.org/wiki/Packaging/Guidelines
- Maximum RPM: http://www.rpm.org/max-rpm-snapshot/
- Fedora CVS Tree (contains lots of examples) http://cvs.fedoraproject.org/viewcvs/rpms/
- rpmlint: http://rpmlint.zarb.org/cgi-bin/trac.cgi
Thanks

• Some material taken from Tom “spot” Callaway's Red Hat Summit presentation from June 2008