RHEL Packaging
(making life easier with RPM)

Jindřich Nový Ph.D., jnovy@redhat.com
June 26, 2012
Agenda

1. How Red Hat Enterprise Linux is packaged

2. Software Collections (SCLs)
   - Filesystem hierarchy layout
   - How to use SCL
   - SCL meta package
   - SCL scriptlets

3. Software collection packaging
   - SCL packaging macros
   - Conversion How-to
   - Why a SCL meta-package is needed?
   - Special cases when packaging a SCL
Section 1

How Red Hat Enterprise Linux is packaged
Software’s way into a distribution

- software release by upstream in a tarball (tar.gz, tar.xz, etc.)
- package maintainer converts the software into a package
- package maintainer builds binary packages for various arches
- package maintainer releases an update/erratum
- end user installs/updates to the latest version
Packaging systems in GNU/Linux distributions

- **RPM** - Red Hat, Mandriva/Mageia, SUSE
- **deb** - Debian, Ubuntu
- **ebuild** - Gentoo
- **compressed files** - Archlinux
- **slackbuilds** - Slackware
- etc.
What an RPM package provides?

- sources
- patches
- software related metadata
  - dependencies
  - software configuration
  - license
  - changelog
  - etc.
- how software is built
- how software is installed
Spec file

Summary: Create a tree of hardlinks
Name: hardlink
Version: 1.0
Release: 14%{?dist}
Epoch: 1
Group: System Environment/Base
URL: http://pkgs.fedoraproject.org/gitweb/?p=hardlink.git
License: GPL+
Source0: hardlink.c
Source1: hardlink.1
Buildroot: %{_tmppath}/%{name}-%{version}-%{release}-root-%{__id_}
Obsoletes: kernel-utils

description
hardlink is used to create a tree of hard links.
It's used by kernel installation to dramatically reduce the
amount of disk space used by each kernel package installed.

prep
%setup -q -c -T
install -pm 644 %{SOURCE0} hardlink.c

build
cc $RPM OPT FLAGS hardlink.c -o hardlink
Limitations of RPM

- no more than one package with the same name installed
  - except multilib and kernel packages
- if one needs to install a newer package incompatible with previous one the whole dependency tree needs to be removed
- uninstallation of a package is sometimes not possible because of wide dependencies
- "dependency hell"
Parallel installability

- GCC toolchain + older versions
- Apache 2.4 + older 2.2
- Perl 5.14 + older versions
- Python 3.2 + older 2.7
- Ruby 1.9.3, Rails 3.2.3 + older versions
- MySQL/PostgreSQL/unixODBC various versions
Section 2

Software Collections (SCLs)
What is a Software Collection (SCL)?

- the aim of the Software Collections is to provide multiple versions of a software in one RHEL
  - the version from collection must not interact with system version
  - system version must not be polluted by collection’s packages
- collection is a system independent package or group of packages
- collections can provide several parallel-installable versions of software
- part of SCL is specific configuration allowing to run applications from SCL environment
SCL highlights and features

- main functionality implemented as set of RPM macros
- 100% under control of RPM packaging system
- compatibility across RHEL versions
  - there is no need to update any of RPM/YUM/RPMBUILD
- minimal spec file modifications to convert an existing package to SCL
- allows to build an unmodified spec as a normal package
- allows to build an unmodified spec into different collection
- solves concurrent SCL update problems
  - there no longer exist update conflicts due to SCL package naming
- inter-SCL dependencies
  - allows to implement multiple levels of SCLs
Software Collections in Fedora/EPEL

- consist of two basic packages

**Runtime utility for running Software Collection applications**

```bash
# yum install scl-utils
```

**Build macros to build Software Collections**

```bash
# yum install scl-utils-build
```

- present in Fedora 15, 16, 17, EPEL5, EPEL6
SCL filesystem hierarchy

```
Filesystem hierarchy layout

/opt/rh/ - configurable via %_scl_prefix
Collection1/
  <arch>/
    root/
    enable
    <Collection1 scriptlets>
  Collection2/
    ...
```
How to enable a SCL?

- scl tool is used to do it for us

**Tool synopsis**

```
$ scl <action> [<SCL1>, <SCL2> ...] <command>
```

**Example of scl tool invocation**

```
$ scl enable example_scl 'perl --version'
```

- it is possible to run shell with SCL enable, after Ctrl-D we are back in untouched system environment
- one can use a wrapper script to simplify execution of a SCL application
SCL packaging layout

- SCL meta package
  - **scl_name** - main SCL package shipping base package set
  - **scl_name-runtime** - package shipping scriptlets and owns SCL filesystem
  - **scl_name-build** - package shipping SCL build configuration (not mandatory)
- SCL packages
  - **scl_name_pkgname** - SCL namespaced and relocated packages
What is SCL scriptlet?

- a simple shell script that changes current environment to prefer SCL package set over a system package set
- currently only `enable` scriptlet is required
- `scl` tool is an interface to use these scriptlets
Section 3
Software collection packaging
How a system and SCL package build differ?

**Normal system package local build**

```
$ rpmbuild -bb package.spec
```

**SCL package local build**

```
$ rpmbuild -bb package.spec --define 'scl <name>'
```
What SCL packaging macro set does?

- relocates file hierarchy to SCL-exclusive filesystem
- defines convenience macros for packagers
- defines file ownerships for the main meta package
Which macros to use in SCL environment?

- SCL specific maros usage need to be prefixed with 
  `%{?scl: ... }`
- `%scl_name` - name of the SCL, e.g. `my_collection`
- `%pkg_name` - original package name, e.g. `ruby`
- `%scl_prefix` - SCL prefix, e.g. `/opt/rh`
  - can be redefined
- `%scl_scripts` - where SCL scriptlets are, e.g. `/opt/rh/my_collection`
- `%scl_root` - package root for a SCL, e.g. `/opt/rh/my_collection/root`
Which macros to use in SCL environment?

- all path macros which are not pointing to SCL filesystem are prefixed with \_root:
  - \%\_root\_prefix \Rightarrow /usr
  - \%\_root\_bindir \Rightarrow /usr/bin
  - \%\_root\_datadir \Rightarrow /usr/share
  - \%\_root\_sysconfdir \Rightarrow /etc
  - \%\_root\_includedir \Rightarrow /usr/include
  - ...

RHEL Packaging
How do I convert ordinary spec to SCL?

```%{?scl:scl_package less}
+
  Summary: A text file browser similar to more, but better
  Name: less
+Name: %{?scl_prefix}less
  Version: 4.43
  Release: 1%{?dist}
  License: GPLv3+
@@ -11,6 +13,7 @@
  URL: http://www.greenwoodsoftware.com/less/
  Buildroot: %{_tmppath}/%{name}-%{version}-%{release}-root-%{__id_u} %
  BuildRequires: ncurses-devel pcre-devel autoconf automake libtool
+%{?scl:Requires:%{scl_runtime}}

%description
The less utility is a text file browser that resembles more, but has
@@ -23,7 +26,7 @@
files, and you’ll use it frequently.

%prep
-%setup -q
+{?scl:-n %{pkg_name}-%{version}}
```
How do I convert ordinary spec to SCL?

- `scl` macro definition needs to be added before package preamble:
  - `%{?scl:%scl_package package_name}`
- Name tag needs to be modified to
  - Name: `%{?scl_prefix}package_name`
- All essential SCL packages should be dependent on main meta package:
  - `%{?scl:Requires: %scl_runtime}`
- `%setup` macro needs to deal with different package name in SCL environment:
  - `%setup -q %{?scl:-n %{pkg_name}-%{version}}`
How should I install a SCL?

- SCL is installed via the main meta package named `scl_name` which contains dependencies to basic SCL package set (i.e. no optional packages)
  - `yum install scl_name`
- Every package in SCL depends on `scl_name-runtime` which contains:
  - base filesystem structure
  - SCL scriptlets
  - optional SCL configuration files
Special cases when packaging a SCL

- libraries
  - \%\{_root_sysconfdir\}/ld.so.conf.d/\%\{scl_prefix\}lib.conf
- initscripts
  - \%\{_root_sysconfdir\}/rc.d/\%\{scl_prefix\}service_name
- manpath
  - put MANPATH enablement script to
    \%\{_root_sysconfdir\}/profile.d/\%\{scl_prefix\}manpages.sh
- cronjobs
- logrotate
- locks
- kernel modules
Software Collection feature summary

- provides a way how to install multiple versions of software in parallel
- is used by several deployments in production
- available in Fedora, EPEL, RHEL6.3
References

- **SCL macros and utilities development:**
  https://fedorahosted.org/SoftwareCollections/

- **Packaging guide:**

- **This presentation:**
Questions?

Thanks for listening.
SEE US AT SUMMIT
Visit us at Developer Zone!

FOLLOW US ON TWITTER
twitter.com/#!/RHELdevelop

PLAY US ON YOUTUBE
bit.ly/RHELdevOnYouTube

LEARN & SHARE AT
red.ht/rheldevoug

GIVE US FEEDBACK
RHELdevelop@redhat.com