Building with the Red Hat Package Manager

29 January 2002 - v1.0.3

updated for RPM v4

Dag Wieërs
dag@wieers.com

Overview

- Introduction
  - Philosophy behind RPM building
  - Build environment
  - Spec files
  - Using RPM to build packages
- Anatomy of the Spec file
  - Header-tags and sections
  - Scripts and macros
  - Changelog
  - Sub-packages
- Signing packages
- Real life example integrated in make-system
- Recommended reading

The philosophy behind RPM building

- Building from original sources and additional patches
- Easier to build
- Reproducible builds
- Unattended builds
- Multi-architecture/operating systems support
- Easier for users (to rebuild)
- Easy package signing

Build environment

- Creating the build directory structure
  - /usr/src/redhat/SOURCES %{_sourcedir}
    - contains the original sources, patches and icons
  - /usr/src/redhat/SPECS %{_specdir}
    - contains the spec files used to control the build process
  - /usr/src/redhat/BUILD %{_builddir}
    - contains the source-trees and object-files
  - /usr/src/redhat/RPMS/{i*86,noarch} %{_rpmdir}
    - contains the binary package files created by the build process
  - /usr/src/redhat/SRPMS %{_srcrpmdir}
    - contains the source package files created by the build process
- Or customized

What is a Spec file and what is in a package ?

- A Spec file contains all the information that is needed to build a source and binary package (or sub-packages)
  - It is a simple text-file
  - It has a (well) defined structure (syntax)
  - One Spec file is needed for one source package
  - One source package can be used to build multiple binary packages
- A source package contains:
  - A Spec file
  - One or more source-trees (usually in tar.gz or tar.bz)
  - One or more patches
  - One or more additional files and an icon
  - Build information
- A binary package is discussed in "Using RPM"

A simple example Spec file

- Let's take the logwrapper package
  - The **Preamble:**

```
Name:       logwrapper
Summary:    A wrapper script for logging all activity of a program
Version:    0.1.0
Release:    dag.1
Group:      System/Tools
Copyright:  GPL
URL:        http://dag.wieers.com/home-made/logwrapper/
Packager:   Dag Wieers <dag@wieers.com>
Source:     ftp://dag.wieers.com/home-made/%{name}-%{version}.tar.bz2
BuildRoot:  %{_tmppath}/%{name}-%{version}
Prefix:     %{_prefix}
%description
logwrapper is a tool that is able to log all activity of a program.
It will log the date, user and full command together with its output.
```

A simple example Spec file

- The **%prep**, **%build, %install** and **%clean** scripts:

```
%prep
%setup

%build
%configure
make

%install
%makeinstall

%clean
rm -rf %{buildroot}
```

- Now you can start the build process:
  - **rpm -ba logwrapper.spec**

Using RPM to build packages

- **rpm -b** (rpm -t)
  - -vv display debugging information
  - -test create, save build scripts for review
  - -p execute %prep
  - -c execute %prep, %build
  - -i execute %prep, %build, %install
  - -b execute %prep, %build, %install (bin)
  - -a execute %prep, %build, %install (bin, src)
  - -l check %files list
  - --clean clean up after build
  - --buildroot execute %install in alternate buildroot
  - --target target platform (%arch-%os)
- **rpm --rebuild logwrapper-0.1.0-dag.1.src.rpm**

Inside the Spec file

- Comments
  - # This is a spec file for logwrapper
- Package naming tags
  - Name, Version, Release
- Descriptive tags
  - Summary
  - Copyright, License
  - Distribution, Group (GROUP in docs)
  - Icon (gif or xpm)
  - Vendor, Packager
  - URL
- Description section
  - %description

Inside the Spec file

- Architecture and operating system-specific tags
  - Buildarch
  - Excludearch, Exclusivearch, Exclusiveos
- Directory-related tags
  - Prefix, Buildroot
- Source and patch tags
  - Source, Patch

Install and build dependency tags

- Normally dependencies and builddependencies are automatically generated.
- The dep script will add package dependencies to file dependencies, if possible.
- Install dependency tags
  - Provides
  - Prereqs
  - Requires
  - Conflicts
  - Obsoletes
- Build dependency tags
  - BuildPrereqs
  - BuildRequires
  - BuildConflicts

Script variables

- Builtin scriptvariables
  - $RPM_SOURCE_DIR %{_sourcedir}
  - $RPM_BUILD_DIR %{_builddir}
  - $RPM_DOC_DIR
  - $RPM_OPT_FLAGS
  - $RPM_ARCH, RPM_OS
  - $RPM_BUILD_ROOT %{buildroot}
  - $RPM_PACKAGE_NAME %{name}
  - $RPM_PACKAGE_VERSION %{version}
  - $RPM_PACKAGE_RELEASE %{release}
  - %{_tmppath}
  - %{_topdir}
  - %{_rpmtopdir}
  - %{_specdir}
  - %{_rpmdir}
  - %{_srcrpmdir}

Script sections and others

- Build-time script sections
  - %prep, %build, %install, %clean,
- Install/erase-time script sections
  - %pre, %post, %preun, %postun
- Other sections
  - %files, %changelog

Macros

- Macro's
  - %define, %undefine
- Builtin macros
  - %trace, %dump, %{echo:...}, %{warn:...}, %{error:...}, ...
- Directory-related macros
  - %buildroot, %_tmppath,
- Macro analogues of autoconf variables
  - %_prefix, %_bindir, %_libdir, %_mandir, %_datadir, %_sysconfdir, %_localstatedir, ...

Build-time builtin macros

- %configure, %makeinstall
- %setup
  - -n set name of build directory
  - -c create directory (and cd) before unpacking
  - -D do not delete directory before unpacking
  - -T do not perform default archive unpacking
  - -b/-a unpack before or after changing
  - -q do it quietly
- %patch
  - -p strip leading slashes and directories
  - -b set the backup file extension
  - -E remove empty output files

File list and sub-packages

- The %files list
  - %defattr
  - %doc
  - %config (noreplace)
  - %attr
- Directory-related directives
  - %docdir, %dir
- Sub-packages
  - %package
- Conditional
  - %ifarch, %ifnarch, %ifos, %ifnos, %else, %endif

Changelog

- %changelog
  - Has a specific format TODO

Signing packages

- Signing macros
  - %_signature signature type
  - %_pgpbin pgp executable
  - %_pgp_name pgp signature name
  - %_pgp_path pgp key ring path
- Easier to specify these into your custom macros-file

Customizing the build environment

- Several config-files allow to change the behaviour
  - /usr/lib/rpm/macros
  - /usr/lib/rpm/%{target}/macros
  - /etc/rpm/macros
  - /etc/rpm/%{target}/macros
  - ~/.rpmmacros
- Some variables are specific to a packager
  - %packager, %distribution, %vendor, %_signature, %_gpg_name
- Some variables allow to override the default locations
  - %_topdir, %_rpmtopdir, %_builddir, %_rpmdir, %_sourcedir, %_specdir, %_srcrpmdir, %_tmppath

Transaction rollbacks

- Method for maintaining sets of packages that should be applied sequentially
- Done by defining the "Transaction set"
- Packages in the same Transaction set use a unique identifier, the "Transaction ID".
- Added since RPM v4.0.3, few documentation available
- But not everything in place to be able to do automatic rollbacks (apply/commit/undo)

Recommended reading

- /usr/share/doc/rpm-%{version}
  - CHANGES, GROUPS, builddependencies, buildroot, dependencies, format, macros, multiplebuilds, queryformat, relocatable, signatures, spec, triggers
- http://www.rpm.org/
  - RPM project with all necessary links to tools and documentation
- http://dag.wieers.com/howto/rpm/
  - The most recent version of this presentation