# Creating RPMs for SUSE® LINUX

Paul MacKay
Software Engineer
Novell, Inc.

Novell®

# Introduction

Consumers want…

- A consistent way to install/upgrade/uninstall applications. They also want to:
  - Query the system for installed applications
  - Preserve system/environment integrity
  - . . .

Developers want…

- A consistent way to build applications where:
  - Packages are built in an isolated environment
  - Build dependencies are resolved automatically
  - A single build environment for various hardware architectures





Novell.

# And the answer is...

RPMs built using SUSE's 'build' utility





Novell.

# Why are you here?

What are RPMs?

How do you create RPMs?

How do you create RPMs using SUSE's 'build' utility?





Novell.

# RPM Overview

# RPM

"RPM is the RPM Package Manager. It is an open packaging system available for anyone to use. It allows users to take source code for new software and package it into source and binary form such that binaries can be easily installed and tracked and source can be rebuilt easily. It also maintains a database of all packages and their files that can be used for verifying packages and querying for information about files and/or packages."*

* See http://www.rpm.org





Novell.

# RPM Industry Acceptance

RPM is the most widely used software package format for Linux Distributions.

- Used by 8 of the top 10 distributions, including SUSE
- Distributions such as Debian, that do not use RPM, have tools to consume RPMs
- Required packaging format for Linux Standard Base (LSB) compliant applications





Novell.

# RPM

RPM is:

A file format

A set of tools to manage rpm files

A database of information about installed applications





Novell.

# RPM File Format

The RPM file format consists of the following sections:

| | |
|---|---|
| Lead | - Used to identify the package |
| Signature | – Used to verify the package |
| Header | - package's name, version, and file list |
| Payload | - GNU zip compressed cpio archive |





Novell.

# RPM Commands

RPM has options to

- Install a package
- Upgrade a package
- Remove a package
- List installed packages
- Display information about an installed package
- List all the files contained in a package
- List the packages on which a package depends
- Display what package provides a specific capability
- Display which packages require a specific capability
- Verify packages and dependencies before installation
- . . .





Novell.

# Examples Using RPM

Install:

```
rpm -i myapp.rpm
```

Erase:

```
rpm -e myapp.rpm
```

Upgrade:

```
rpm -U myapp.rpm
```

Verify:

```
rpm -V myapp.rpm
```

Query:

```
rpm -q myapp
```





Novell.

# RPM Database

The RPM Database is located in:

/var/lib/rpm

When the RPM database becomes corrupt (hopefully this will not ever happen) you can rebuild the RPM database via:

```
rpm --rebuilddb
```





Novell.

# Building an RPM Package

# Where to start?

The best place to start is to analyze existing source RPM files

A simple RPM to start with is the GNU Hello world package

```
hello-<latest-version>-src.rpm
```





Novell.

# What is in a Source RPM?

A source RPM contains a spec file, a tarball containing the source files and perhaps some patch files

To look at the content of a source RPM use `rpm2cpio` and `cpio`

For example:

```
rpm2cpio hello-2.1.1-309.2.src.i386.rpm > hello.cpio
```

```
cpio -id < hello.cpio
```





Novell.

# Where are Source RPMs Installed?

A source RPM it is installed under "`/usr/src/packages`".

If you install `hello<latest-version>.rpm` the following files are installed:

```
/usr/src/packages/SOURCES

/usr/src/packages/SOURCES/hello-2.1.1.tar.gz

/usr/src/packages/SPECS/hello.spec
```





Novell.

# RPM Package Building Directories

/usr/src/packages/BUILD

Area where the source is unpacked and built

/usr/src/packages/SOURCES

Contains the original source and patch files

/usr/src/packages/SPECS

Contains the package's spec file

/usr/src/packages/RPMS

Contains the created binary RPM

/usr/src/packages/SRPMS

Contains the created source RPM





Novell.

# Build the Existing Source RPM

To build an RPM in `/usr/src/packages` you need to login as root

Change directory to the `/usr/src/packages/SPECS` directory and type:

```
rpmbuild -ba hello.spec
```





Novell.

# rpmbuild

It is used to build both binary and source packages
rpmbuild major options:

-bp
Execute %prep section
-bc
Execute %build section
-bi
Execute %install section
-bs
Build source package
-bb
Build binary package
-ba
Build both source and binary packages

# Build the Existing Source RPM?

If you are lucky it may succeed. However, most of the time it does not

What are some of the issues in creating RPMs using only the `rpmbuild` Tool?





Novell.

# Issues in Creating RPMs

Using `rpmbuild` to create RPMs poses several issues for the developer/packager:

Build dependencies must be managed by hand

- The installation of required packages to build the package is often determined by trail and error (run `configure` and see what happens)

Not building in an isolated environment

- The current system may have tools and libraries that are not compatible or reflect the target installation environment

Easy to pollute the current environment

- Testing the installation of binary RPMs may affect the current system because of RPM dependencies





Novell.

# How to use SUSE's 'build' Utility

## to create RPMs

# Creating RPMs for SUSE Linux

Make necessary modifications to the spec file

- Use “standard” macros
- Add a BuildRequires section
- Obtain an image of the given SUSE product
  - The default is to use a DVD

Use SUSE's `build` utility to isolate and manage the RPM creation process





Novell.

# RPM Spec File
## A Quick Review

The RPM spec file contains the following sections:

- The “preamble”
- %prep
- %build
- %install
- %pre
- %post
- %files
- %clean
- %changelog





Novell.

# The Preamble

Summary

- One-line description of the package

Name, Version, Release

- Name and version of the package. Release is the build number of the rpm itself

License

- Indication of the licensing terms

Group

- Identifies the type/classification of software

Source0, Source1, etc.

- Usually a tarball. "`%{name}.%{version}.tar.gz`"

Patch0, Patch1, etc.

- Identifies the patch files to use on the source files





Novell.

# RPM Spec File (cont.)

Various tags:

%description

A longer description of the package

%prep

Installs the source files to a build area.The % patch can be used to run the patch program

%build

Commands to compile the source (e.g. configure, make etc.)

%install

Package installation commands





Novell.

# RPM Spec File (cont.)

%pre

Pre-installation actions (e.g. check for existing files/directories etc.)

%post

Post-installation actions (e.g. call `ldconfig` etc.)

%clean

Cleanup actions after the rpm build process.

%changelog

Used in later versions of RPM to record the changes to the rpm spec file





Novell.

# Typical Processing
## Though an RPM ‘spec’ File

1. Spec file is read and parsed
2. %prep section is run
   - Source code is unpacked and patches are applied
3. %build section is run
   - Source code is compiled
4. %pre
   - Pre-installation steps
5. %install section is run
   - Install the package in a 'dummy' area
6. %post
   - Any post-installation steps
7. %files section is run
   - List of files are read and gathered from the 'dummy' area and the binary and source RPM files are created
8. %clean
   - Remove temporary build directories that may have been created

# Now What?

We now understand spec files and how rpmbuild processes them

How do I create RPMs for SUSE LINUX in a clean and efficient way?





Novell.

# SUSE's Build Tool

`build` is a tool used to create RPMs in an isolated environment

`build` will install a minimal SUSE LINUX environment into a given directory, `chroot` to this created Linux environment, and compile the package





Novell.

# SUSE's Build Tool (cont.)

`build` searches the spec file for an extra line "`BuildRequires: `"

If such a line is found, all specified RPMs after the "`BuildRequires: `" tag are installed into the temporary Linux environment*. Otherwise a default set of packages will be installed

*Note: `build` does not automatically resolve missing dependencies. The specified rpms on the "`BuildRequires:`" line must be sufficient for the build to succeed





Novell.

# Adding the BuildRequires Section

The "`BuildRequires:` " section is a list RPM package names separated by a space all on a single line

For example:

```
BuildRequires: aaa_base acl attr bash bind-utils bison
bzip2 coreutils cpio cpp cracklib cvs cyrus-sasl db devs
diffutils e2fsprogs file filesystem fillup findutils
flex gawk gdbm-devel glibc glibc-devel glibc-locale gpm
grep groff gzip info insserv kbd less libacl libattr
libgcc libselinux libstdc++ libxcrypt m4 make man mktemp
module-init-tools ncurses ncurses-devel net-tools netcfg
openldap2-client openssl pam pam-modules patch
permissions popt procinfo procps
```

# Installing Required RPMs

By default, `build` installs the RPMs specified in the `BuildRequires` section from "`/media/dvd/suse`"

To change the default location of RPMs, use the shell environment variable `BUILD_RPMS` to specify an alternative location





Novell.

# Demonstration Using build

# What Have You Learned?

A better understanding of RPMs

What RPMs are made of

What are RPM spec files and how are they processed

How to use SUSE's `build` utility to make creating RPMs for SUSE LINUX in a clean and efficient way





Novell.

# Questions and Answers

# References

The 'Official' site:

- http://www.rpm.org

Software Packaging with RPM:

- http://www.linux-mag.com/2004-02/compile_01.html

Building RPM Packages

- http://www-uxsup.csx.cam.ac.uk/talks/rpmbuild/rpmbuild.pdf

RPM 101

- http://linuxvm.org/present/SHARE99/S9372NFa.pdf






Novell.

Novell®

Novell