Workshop:
Building Good Fedora Packages

Pavel Raiskup
<praiskup@redhat.com>

Miroslav Suchý
<msuchy@redhat.com>

Slides originally created by
Tom 'spot' Callaway

Things to do to prepare your environment

- Packages that you will need installed
  - fedora-packager, rpm-build, dnf (or yum on RHEL <8), rpmdevtools, rpmlint, patch
- Get a copy of the files we will be working with:
- https://praiskup.fedorapeople.org/courses/packaging/
- Use the “rpmdev-setuptree” command from the “rpmdevtools” package to create ~/rpmbuild directory tree

Workshop Overview

- Topic: intro to RPM, create a simple RPM package from scratch
- Assumptions
  - You know how to use a text editor (no matter which one)
  - You know how to manually build “normal” software for Linux
  - You know what is a patch and how to apply it
  - You have some familiarity with installing RPM packages
- Limitations
  - This workshop covers a simple piece of software
  - Most packages will be a little more complicated
    - Some will be a LOT more complicated
- Information
  - Feel free to ask questions! Break?

fedora

Objectives - RPM Package Management

- Why to use RPM packages?
- Can package be problematic?
- What is source and binary package?
- Difference between file name and package name
- Basic commands

Packaging as a standard (aka, why package at all?)

- Auditing software usage
  - What, where?
- Version control
- Kickstart integration (anaconda)
- Minimizes risk
  - Security
  - Rogue Applications
  - Licensing
  - Trusted provider
- So you see why `make install` is just not enough...

Problems with RPMs? No.

- RPM is misunderstood
- Works extremely well
- Package creation is easier than you think
- Easy to install...
- Easy to remove...
  - ... with good packages!
- You can write good programs and bad programs
- And you can write good packages and bad packages

Crash course in RPM Usage

- Binary (built) Package pkg-workshop-1-0.x86_64.rpm
  - File name is different from package name
- Install packages with file name
  - rpm -ivh pkg-workshop-1-0.x86_64.rpm
  - i for install, v for verbose, h for process hash
- Query installed package with package name
  - rpm -ql pkg-workshop
  - q for query, l for list files
- Remove package with package name
  - rpm -eh pkg-workshop
  - e for erase
- List all installed packages in RPM database
  - rpm -qa
  - rpm -qi <pkgname> (info about package)
- There's DNF/YUM on top of RPM databse, but that's not the topic for today

fedora

Source RPM Overview

- Source Package (pkg-workshop-1-0.src.rpm)
  - SRPMs contain sources/components/spec file used to generate binary RPM packages
  - Headers (/bin/mc can show)
- Install SRPM package with SRPM file name
  - rpm -ivh pkg-workshop-1-0.src.rpm
  - i for install, v for verbose, h for process hash
  - Source packages just install source into defined source directory
    - Red Hat default: ~/rpmbuild/SOURCES
- SRPMs do not go into the RPM database
- Remove installed SRPM with spec file name
  - rpmbuild --rmsource --rmspec pkg-workshop.spec

More Source RPM Overview

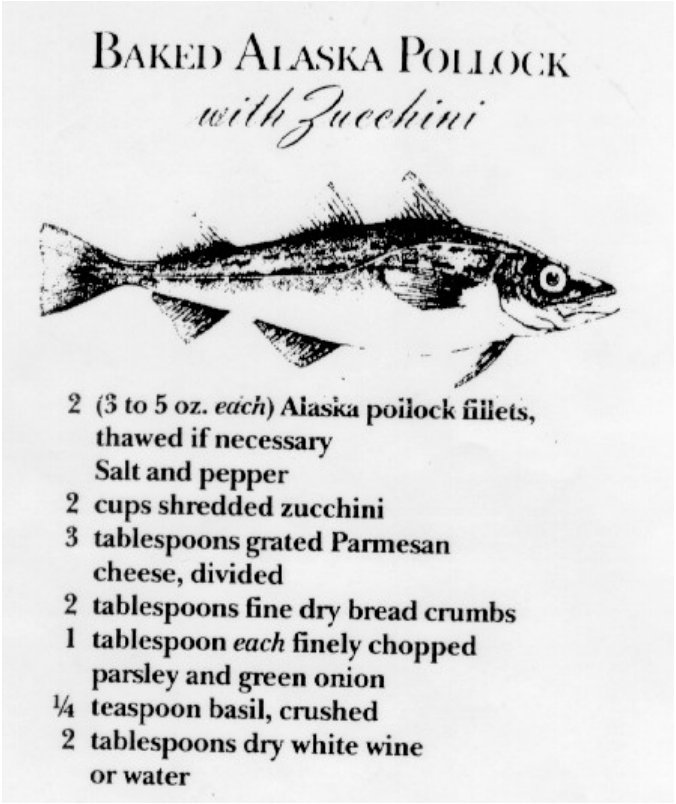
- Making a binary rpm from SRPM:
  - rpmbuild --rebuild pkg-workshop-1-0.src.rpm
- Making a binary rpm from spec file
  - rpmbuild -ba pkg-workshop.spec
  - -b for build, -a for all packages, src and bin
- Making a patched source tree from spec file
  - rpmbuild -bp goldfish.spec
  - -b for build, -p for %prep only
- Patched source trees go into the builddir
  - Red Hat default is ~/rpmbuild/BUILD

Objectives - Spec file

- What is a SPEC file
- Generators
- Basic syntax:
  - RPM Macros
  - Comments
- Best practices

Cooking with Spec Files

- It's a simple text file
- Think of a spec file as a recipe
- Lists the contents of the RPMS
- Describes the process to build, install the sources
- Required to make packages
- Very similar to shell script
- Sections/stages:
  - Preamble
  - Setup
  - Build
  - Install
  - Clean
  - Files
  - Changelog

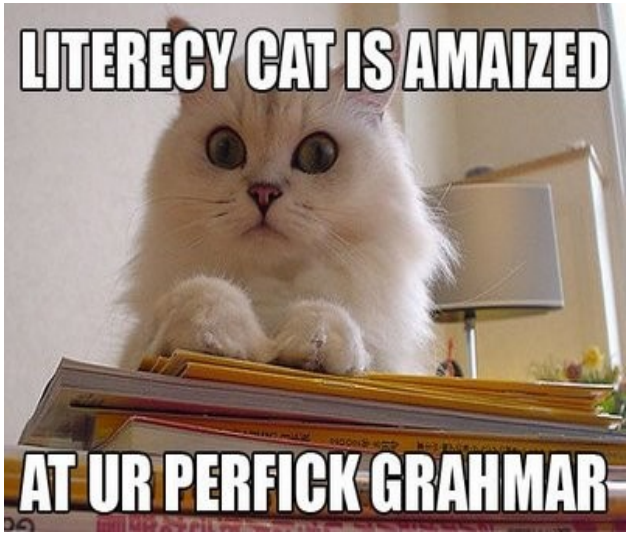


fedora

Common mistakes new packagers make

- Spec file generators
  - Remember, functional is not the same as good.
- Packaging pre-built binaries, not building from source.
  - Not always possible, but you shouldn't start here if you can help it.
  - Not permitted in Fedora
- Disabling check for unpackaged files
  - This is a recipe for disaster.

RPM Macros



- Just like variables in shell scripting
  - They can act as integers or strings, but its easier to always treat them as strings.
- Macro formats
  - %{foo} and %foo
  - some macros have shell variable variant, e.g. $RPM_BUILD_ROOT vs. %buildroot, they hold the same value, but for your sanity (and guidelines), you should consistently use one type of macro in a spec file.
- Many common macros come predefined
  - **rpm --showrc** will show you all the defined macros, number explanation
  - **rpm --eval %macroname** will show you what a specific macro evaluates to
  - Most system macros begin with an _ (e.g. %{_bindir})

RPM Comments

- To add a comment to your RPM spec file, simply **start** a new line with a # symbol. Feel free to do this as we go, to take notes for yourself. It never hurts to explain in a comment why you did something, and it may save a bit of your sanity later on.
- For example:
  # I have to delete this file, or else it will not build properly.
  rm -f foo/bar/broken.c
- RPM ignores comment lines entirely.
  - Well, to be fair, **this isn't true**, sometimes if you # comment out a macro definition, it will see it and evaluate it anyways. To comment out a macro definition, use two %% instead of just one:

    Before: # %configure
    After: # %%configure

Custom rpm macros ~/.rpmmacros

- Syntax: %macroname value
- Overrides for system-defined macros /usr/lib/rpm/
- Can be overwritten directly in spec file
  - Syntaxe: %global macroname value
- You can make your own macros here, be careful! (why?)
  - .. so, don't use them in spec file
- example with %_smp_mflags

Fedora Packaging Guidelines

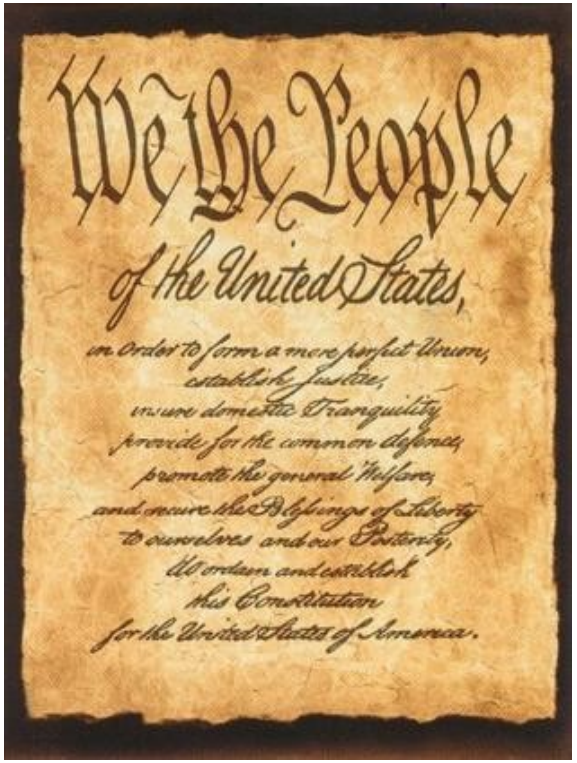
- Intended to document a set of “best practices”
- Living document, constantly being amended and improved
- Exceptions are possible
  - Common exception cases are usually documented
  - If you can justify doing something differently, it is usually permissible, although, it may need to be approved by the Fedora Engineering Steering Committee (FESCo)
  - Use common sense, but when in doubt, defer to the guidelines
- https://docs.fedoraproject.org/en-US/packaging-guidelines/

Objectives - real-life spec - preamble

- What is a preamble?
- What tags can be defined in a preamble?
- Write simple preamble section.

Understanding the Spec File: Preamble

- Initial section
- mostly metadata (no scripts)
  - defines output of `rpm -qi tar`
- Defines package characteristics
  - Name/Version/Group/License
  - Release tracks build changes
  - Sources/Patches
  - Requirements, for
    - Build & Install
  - Summary/Description
  - Custom macro definitions



fedora

Workshop: Creating a new spec file

- A spec file is simply a text file
- RPM expects spec files to be in ~/rpmbuild/SPECS/
  - Technically, it doesn't care, but for sanity, lets just keep them there.
- Go ahead and open a new text file called ~/rpmbuild/SPECS/enum.spec
- Some editors will generate a spec template when you open a new spec file (notably, vim before rhbz#1724126, now it’s in /usr/share/vim/vimfiles/template.spec)
- rpmdev-newspec

Workshop : Preamble Items

Here is a blank preamble section, this is where we will start our package!

Name:
Version:
Release:
Summary:
License:
~~Group:~~
URL:
Source0:

%description

Workshop : Name

- First, lets fill in the name of our package, which is “enum”.

```
Name:       enum
Version:
Release:
Summary:
License:
URL:
Source0:

%description
```

- Note: Either use spaces or tabs to separate your fields. Doesn't matter which one as long as you are consistent.

Workshop : Version

- Next, lets fill in the version of our package, which is “1.1”.

```
Name:       enum
Version:        1.1
Release:
Summary:
License:
URL:
Source0:

%description
```

- Hopefully, it should be obvious where I got the version from. :)
  Sane upstreams will put the version in the source tarball name.
  Other code may require digging.

Workshop : Release

- The Release field is where you track your package builds, starting from 1 (guidelines), and incrementing by 1 each time you make a change.

```
Name:       enum
Version:        1.1
Release:        1%{?dist}
Summary:
License:
URL:
Source0:

%description
```

- If your package is a pre or a post release, there are special rules for handling Version and Release, see:
  https://docs.fedoraproject.org/en-US/packaging-guidelines/Versioning/
- Use /bin/rpmdev-vercmp if you are in doubt!

Wait, what is that %{?dist} thing?

- It is a macro!
- It is there to add an identifier (or dist) tag to the end of the release field. For humans!
- The ? means “if defined, use it, if not defined, evaluate to nothing”
- So, if your release field is:

Release: 1%{?dist}

- Then, it evaluates to “1.fc33” on Fedora 33, and “1.el8” on RHEL 8.
- You can see more information on the Dist Tag here:
- https://docs.fedoraproject.org/en-US/packaging-guidelines/DistTag/

Workshop : Summary

- Summary is a single sentence describing what the package does. It does not end in a period, and is no longer than 80 characters.

```
Name:       enum
Version:        1.1
Release:        1%{?dist}
Summary:        Seq- and jot-like enumerator
License:
URL:
Source0:

%description
```

- NOTE: We'll fill in a longer description of the package in %description

Workshop : License

- The License tag is where we put the short license identifier (or identifiers) that reflect the license(s) of files that are built and included in this package. It is easiest to determine the correct license once we have an unpacked source tree, so we'll put “TODO” in this field, and fix it later.

```
Name:       enum
Version:        1.1
Release:        1%{?dist}
Summary:        Seq- and jot-like enumerator
License:        TODO
URL:
Source0:

%description
```

Workshop : URL

- The URL tag is a link to the software homepage.

```
Name:       enum
Version:        1.1
Release:        1%{?dist}
Summary:        Seq- and jot-like enumerator
License:        TODO
URL:        https://fedorahosted.org/enum
Source0:

%description
```

Workshop : Source0

- The Source0 tag tells the rpm what source file to use. You can have multiple Source# entries, if you need them. Put the full upstream URL where you downloaded the file.

```
Name:           enum
Version:        1.1
Release:        1%{?dist}
Summary:            Seq- and jot-like enumerator
License:        TODO
URL:            https://fedorahosted.org/enum
Source0:            https://fedorahosted.org/releases/e/n/enum/%{name}-%
{version}.tar.bz2

%description
```

- rpmbuild takes the basename. Full URL is mostly for humans.

Wait, why did you use macros there?

- You should have noticed that I used
  %{name} evaluates to whatever we have set as Name.
- %{version} evaluates to whatever we have set as Version.
- By doing this, it means that we should not have to change the Source0 line as new versions release (or if upstream changes the name).
- In fact, all of the fields in the preamble are defined as macros, in the exact same way!
- In your spec files, you should try to use these macros whenever possible.

Copy sources to ~/rpmbuild

- This is a great time to copy (not unpack) the enum-1.1.tar.bz2 source into ~/rpmbuild/SOURCES/

Workshop : %description

- %description indicates to RPM that you are entering a block of text which describes the package.
- This can be multiple lines, but should be concise and describe the functionality of the package.
- No line in the %description can be longer than 80 characters and it must end with a period.
- Try not to simply repeat the summary.

```
%description
Utility enum enumerates values (numbers) between two values, possibly
further adjusted by a step and/or a count, all given on the command line.
Before printing, values are passed through a formatter. Very fine control
over input interpretation and output is possible.
```

Workshop : Complete Preamble

Name: enum
Version: 1.1
Release: 1%{?dist}
Summary: Seq- and jot-like enumerator

License: TODO
URL: https://fedorahosted.org/enum
Source0: https://fedorahosted.org/releases/e/n/enum/%{name}-%{version}.tar.bz2

%description
Utility enum enumerates values (numbers) between two values, possibly further adjusted by a step and/or a count, all given on the command line. Before printing, values are passed through a formatter. Very fine control over input interpretation and output is possible.

Preamble: Explicit Requires

- Requires – This lists any packages which we know are necessary to be present on the system to **run** the software in our package, once it is installed.
- RPM usually does a very good job of autodetecting dependencies and adding them for you especially when the software is in C, C++, Perl, etc.
- Think twice if you need to be explicit.
- Requires can be versioned:
  - Requires: bar >= 2.0
  - Requires: bar < 10.0.0

Preamble: BuildRequires

- BuildRequires – This lists the packages which need to be present to build the software.
- Most packages have at least one BuildRequires:

```
BuildRequires:  make
```

- We will see later if our package need some BuildRequires
- You can list as many packages as BuildRequires as you need, although, you should try to avoid redundant items. Also, BuildRequires can be versioned:

```
BuildRequires:  bar >= 2.0
```

Preamble: Patch

- Patch0 – If you need to apply a patch to the software being packaged, you can add a numbered patch entry here:

```
Patch0:     enum-1.1-use-putchar.patch
```

- Patch can be a full URL as well Source<N>, if it makes sense

%if condition

- How to handle different OSes with one spec? Use macros and conditions:

```
## BUGs!
%if 0%{?rhel} <= 6 || 0%{?fedora} < 17
Requires: ruby(abi) = 1.8
%else
Requires: ruby(release)
%endif
```

Preamble: Explicit Provides

- Provides: tell rpm that this package provides something else. E.g. Httpd provides webserver
- Note: boolean dependencies

Objectives - real spec - continued

In this chapter you will:

- learn about main sections of SPEC file: prep, build, install, files and changelog
- learn about licenses
- learn about scriptlets
- how to build the binary package from SPEC

Understanding the Spec: Setup

- Source tree is generated in ~/rpmbuild/BUILD
- Sources unpacked here
- Patches applied
- Any pre-build actions
- Example of a %setup stage:

  %prep
  %setup -q
  %patch0 -p1

- modern alternative: %autosetup -p1

Workshop : Prep & Setup

- First, we need to add a %prep line to tell rpm that we're in the %prep phase.
- %setup is a very powerful (and complicated) “macro” that is included with RPM. It is used to unpack Source# files, into ~/rpmbuild/BUILD/
  So, below our %description text, we'll add:

```
%prep
%setup -q
```

- The -q option tells %setup to unpack the Source file quietly. If you want to see what is happening here, you can omit it, but it usually is a good thing to keep the build logs small and easy to read.
- By default, %setup unpacks Source0 only. It is possible to use %setup to unpack multiple Source# files at once, for more details on complicated use cases, see Maximum RPM docs

Prep: Other items

- %patch0 – If we had a Patch0 entry in our spec, we would apply it here with the matching %patch# macro. Some common options that are good to know:
  - -p# - the patch level (how many directories deep does this patch apply)
  - -b .foo – a patch suffix, appended to the original files before patching. This is very useful when you need to update or change a patch.
- So, for example, a spec with
  Patch0: enum-1.1-use-putchar.patch
  would also need:
  %patch0 -p0 -b .use-putchar
- Note that patch patch use fuzzy 0!

Workshop : %prep

- Here's what our spec looks like now (try rpmbuild -bp enum.spec)

```
Name: enum
Version: 1.1
Release: 1%{?dist}
Summary: Seq- and jot-like enumerator
License: TODO
URL:     https://fedorahosted.org/enum
Source0: https://fedorahosted.org/releases/e/n/enum/%{name}-%{version}.tar.bz2
Patch0: enum-1.1-use-putchar.patch

%description
Utility enum enumerates values (numbers) between two values, possibly
further adjusted by a step and/or a count, all given on the command line.
Before printing, values are passed through a formatter. Very fine control
over input interpretation and output is possible.

%prep
%setup -q
%patch0 -p0 -b .use-putchar
```

Understanding the Spec: Build

- Binary components created (“compiled”) within ~/rpmbuild/BUILD
- Use the included %configure macro for good defaults
- Example of a %build section

  ```
  %build
  %configure
  %make_build
  ```

- If your package uses “scons”, “cmake”, alter accordingly (we’ll show later how to find out).

Workshop : Build

- Here's what our spec looks like now.

```
...
%prep
%setup -q


%build
%configure
%make_build
```

- command `rpmbuild -bc enum.spec` should work now, fix bugs!

Understanding the Spec: Install

- Creates buildroot, ~/rpmbuild/BUILDROOT - a directory where all build artifacts are installed on the final path. This can include all non-compiled files which should be in final package: documentation, examples, man pages, data files.
- Lays out filesystem structure
  - mkdir -p %{buildroot}/var
  - mkdir -p %{buildroot}/%{_bindir}
- Puts built files in buildroot
  - cp -a result-of-make/my-application %{buildroot}/%{_bindir}/my-application
- Cleans up unwanted installed files (if needed):

```
%make_install  # installs too much stuff
rm -rf %{buildroot}/{_datadir}/non-free/
```

Workshop – Fixing our Install Section

- Our build section is done now. Now, we need to fix up our %install section. We know that we need to use “make install” to install... **try it** now.
- ... but RPM doesn't want to install the files into their positions on /. We need to install the files into our BuildRoot, so that RPM can collect them and package them up in the binary rpm file.
- Here's how you do this. Add this line below %install:

~~make DESTDIR=%{buildroot} install # old variant~~

- %make_install

- That's all! The DESTDIR variable tells make to install into our %{buildroot}.
- Some sloppy software Makefiles may not support DESTDIR, if you come across one of these, you should try to add support to the Makefile. Feel free to ask on #fedora-devel or devel@lists.fedoraproject.org for help with this.

fedora

Workshop : Install

BuildRequires: asciidoc

...

%build

%configure ~~--disable-doc-rebuild~~

~~make %{?_smp_mflags}~~

%make_build

%install

~~rm -rf $RPM_BUILD_ROOT # don't do this nowadays~~

~~make install DESTDIR=%{buildroot} # old variant~~

%make_install

Understanding the Spec: %files

- Files: List of package contents
  - If it is not in %files, it is not in the package.
  - RPM WILL complain about unpackaged files.

- We'll come back to this section at the end, when we know what to put in it. For now, lets just define the section, by adding the %files line below our %install section:

  %files

Understanding the Spec: Changelog

- Used to track package changes
- Not intended to replace source code Changelog
- Provides explanation for package **users**, audit trail
- Update on EVERY change
- Example of Changelog section:

%changelog
* Mon Jun 2 2008 Tom “spot” Callaway <tcallawa@redhat.com> 1.1-3
- minor example changes

* Mon Apr 16 2007 Tom “spot” Callaway <tcallawa@redhat.com> 1.1-2
- update example package

* Sun May 14 2006 Tom “spot” Callaway <tcallawa@redhat.com> 1.1-1
- initial package

Workshop : Changelog

- Below our %files section, as the last item in the spec, add a line for changelog

  %changelog

- Then, lets add our first changelog entry, in the proper layout:

  * Fri Sep 21 2012 Your Name Here <youremail@here> - 1.1-1

  - New package for Fedora

- You may use “rpmdev-bumpspec enum.spec” to do the most hard work for you.

Workshop – Spec File Status

```
Name: enum
Version: 1.1
Release: 1%{?dist}
Summary: Seq- and jot-like enumerator
License: TODO
URL:     https://fedorahosted.org/enum
Source0: https://fedorahosted.org/releases/e/n/enum/%{name}-%{version}.tar.bz2

%description
Utility enum enumerates values (numbers) between two values, possibly
....

%prep
%setup -q

%build
%configure
%make_build

%install
%make_install

%files

%changelog
* Fri Sep 21 2012 Your Name Here <youremail@here> - 1.1-1
- New package for Fedora
```

Workshop – Building our source tree

- Now, we can use that source tree to help us fill in the blanks we left earlier - License: TODO
- At this point, we can use our spec to build our source tree and take a look around it.
- Save your spec file, and run (as a normal user):
- rpmbuild -bp ~/rpmbuild/SPECS/enum.spec
- The -bp option tells rpmbuild to run through the %prep stage, then stop.
- It should complete without errors, and you should see a new directory in ~/rpmbuild/BUILD/enum-1.1/

Workshop - Licensing

- As a responsible Fedora packager, it is important that you do the licensing correct for your package. Here are some general steps to follow:
  - Is there a COPYING or LICENSE file? If so, read it, and remember its file name, because we'll want to include it in our package.
  - Is there a README file? Read it, and look for any mention of “Copyright” or “License”.
  - Look at the actual source files in your text editor. Some code projects will describe their license in the header comments of each source file.
  - Note all licenses that you see, then look them up in the Fedora Licensing chart, found here: https://fedoraproject.org/wiki/Licensing#Software_License_List
  - What license do you find for enum?

Workshop – Licensing Part Two

- enum is licensed under the BSD Licence. In Fedora, the short name identifier for this license is BSD.
- Licensing can be very complicated! When in doubt, feel free to email fedora-legal@lists.fedoraproject.org or legal@fedoraproject.org to get help.
- Now, we need to fix our spec. Open it up in a text editor again, and change the License tag from TODO to BSD
- Also, did you see that file COPYING? We need to make sure they are installed in our package, so we'll add them to our %files list, using a macro called “%license” (formerly %doc was used).
- licensecheck helper
- License changes must be announced on developer list

Workshop – Understanding %doc

- %doc is a special macro that is used to:
  - Mark files as documentation inside an RPM
  - Copy them directly from the **source** tree in ~/rpmbuild/BUILD/enum-1.1/ into the “docdir” for your package, ensuring that your package always has them.
- So, lets add a line to %files for our license texts, so that it now looks like this:

```
%files
%license COPYING
%doc ChangeLog
```

- Once you're done, save your spec file again.

Workshop – How Does Enum Build and Install?

- Now, we need to look in the ~/rpmbuild/BUILD/enum-1.1/ source tree to figure out how to build and install the code.
- Here's a big hint: Many Linux software projects use these commands to build:

```
./configure
make
```

And these commands to install:

```
make install
```

- Some packages will not be so clean. Look for INSTALL or README to describe it, or the project website. When in doubt, ask!

History Lessons

- Before Fedora 12 (and in RHEL 5 or older), it used to be necessary to manually remove the %{buildroot} as the very first step in the %install stage, like this:

  %install
  ~~rm -rf %{buildroot}~~

  With Fedora 12 (or newer) RPM, the %install stage automatically deletes the %{buildroot} for you as the very first step, so this is no longer necessary.

- Also, it used to be necessary to define a %clean section to clean up the %{buildroot} at the end of the package build process, it looked like this:

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

  With Fedora 12+ RPM, this %clean section is handled internally and does not need to be explicitly defined.

History Lessons #2

- In the recent past, there was a mandatory field in the Preamble statement called “BuildRoot” - it is now defined automatically. (EL6+), you can drop such lines:

```
BuildRoot: %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)
```

Workshop - Status

- Okay, history lesson over. Your spec should have a fleshed out %build and %install, which look like this:

  see console

- Save your spec file out. There is one thing we're missing from our spec, the list of files to go into %files! I'm about to share a clever trick on how to make it easier.

Workshop – Build our package

- Now, as a normal user run:

```
rpmbuild -ba ~/rpmbuild/SPECS/enum.spec
```

- The -ba options tell rpmbuild to build “all”, source and binary packages.
- This is going to run through %build, then %install ... and then fail, because we don't have a complete %files list specified

Workshop – Build our package

- But, when it fails, it does us a favor, check out the output!

```
RPM build errors:
Installed (but unpackaged) file(s) found:
/usr/bin/enum
/usr/lib/debug/usr/bin/enum-1.1-1.fc33.x86_64.debug
/usr/share/man/man1/enum.1.gz
```

- RPM has just told us what files are missing from the %files list!

Workshop – Adding missing %files

```
%files
%license COPYING
%doc ChangeLog
%{_mandir}/man1/enum.1*
%{_bindir}/enum
```

We skipped the debug file, because once we added the corresponding executable then rpmbuild picks the debug file automatically.

Workshop : Files

- You may also add a “%defattr” line. The %defattr macro tells RPM what to set the default attributes to for any and all files in the %files section. The Fedora default is “(-,root,root,-)”. You do not need to define it again.

  %attr(<file mode>, <user>, <group>, <dir mode>)

- http://ftp.rpm.org/max-rpm/s1-rpm-specref-files-list-directives.html

Workshop – Almost done

- At this point, make sure you have all useful documentation listed in %files as %doc, not just the license text. Look for README and ChangeLog.
- INSTALL is usually not very helpful, do not install it
- That should be it! Look over your spec and make sure you're happy with it, then save it, and run:

```
rpmbuild -ba ~/rpmbuild/SPECS/enum.spec
```

Workshop – Status Check

- At this point, you should now have a enum-1.1-1.src.rpm and one arch rpm: enum-1.1-1.fc*.*.rpm (the Fedora versions and architectures will vary, depending on your system).
- If so, congratulations! You're on your way to really understanding how to make good Fedora packages!
- If it failed, no worries. Take a look at the last few lines that RPM output, and it will probably give you an idea of what to fix. Feel free to ask me for help.
- The next step is to check the package for minor issues, and you use a tool called “rpmlint” for this. Run:

  rpmlint ~/rpmbuild/SRPMS/enum-1.1-1.src.rpm ~/rpmbuild/RPMS/*/enum*.rpm

- If you find any errors, try to correct them in the spec and rebuild. If you do make changes to your spec file, increment your Version and add a new changelog entry at the top of your %changelog section!

Bonus - RPM Scriptlets

- Shell code sections executed at certain times during package installation, %pre, %post, %preun, %postun.
- Scriptlets are non-interactive, no arguments, macros are expanded at package **build time** and baked-into the RPM file
- Remember, RPM DB doesn’t track what is done here
- The best intentions can easily go evil (it is hard to get scriptlets right), but sometimes are necessary (e.g. create system users during package installation)
- More info: RPM packaging guide and Fedora guidelines.

Best Practices

- K.I.S.S.
- Use patches, not rpm hacks
- Avoid scriptlets, minimize wherever possible
  - Leverage triggers instead (fedora docs)
- Use %changelog
- Look at and learn from other Fedora packages
  - Huge tarball with all spec files
- Use macros sanely
  - Be consistent
  - Utilize system macros

Do not build a package as root

- Do NOT EVER build RPMS as root.
  Let me repeat, do NOT EVER build RPMS as root.

%install

%make_install

remove patent protected files

rm -rf $RPM_TYPO_BUILDROO/var/

!!!! THIS WILL REMOVE /var on your system !!!! Why?

Better than Best Practices

- Use rpmlint, fix warnings/errors
- Include configs/scripts as Source files
- Comment!
  - ...but keep it legible
  - Think of the guy who will have to fix your package when you leave.
- Don't ever call /bin/rpm* from inside a spec file.
  - RPM is NOT re-entrant

Good Packages Put You In Control

- Practice makes perfect
- Integration with the Fedora tools makes it easier for users to get and use that software!
- Simplify, standardize, save time and sanity
  - Build once, install many.

Useful Links

- Fedora Packaging Guidelines:
  https://docs.fedoraproject.org/en-US/packaging-guidelines/
  https://fedoraproject.org/wiki/Packaging:ReviewGuidelines
- Maximum RPM:
  http://www.rpm.org/max-rpm-snapshot/
- RPM Packaging Guide:
  https://rpm-packaging-guide.github.io/
- Fedora GIT Tree (contains lots of example specs)
  https://src.fedoraproject.org/
- Fedora packaging mailing list
  https://admin.fedoraproject.org/mailman/listinfo/packaging
- Rpmlint website:
  http://rpmlint.zarb.org/cgi-bin/trac.cgi
- RPM.org documentation
  http://rpm.org/documentation.html