RPM 打包技术与典型 SPEC 文件分析

关键字: rpm spec
RPM 打包技术与典型 SPEC 文件分析

本文分为两部分，第一部分阐述了 rpm 工具的功能以及 rpmbuild 工具，详细的介绍了 spec 文件的书写规则以及关键部分，第二部分对一个典型的 spec 文件做了详细的分析。

为了方便朋友们查看，我找了中文的 rpmbuild 的 MAN 附在最后

一 、rpm 介绍

1. 概述

RPM 全称是 Red Hat Package Manager（Red Hat 包管理器）。几乎所有的 Linux 发行版本都使用这种形式的软件包管理安装、更新和卸载软件。

RPM 是一个开放的软件包管理系统。它工作于 Red Hat Linux 以及其它 Linux 和 UNIX 系统，可被任何人使用。redhat 软件公司鼓励其它厂商来了解 RPM 并在自己的产品中使用它。RPM 的发布基于 GPL 协议。对于最终用户来说，使用 RPM 所提供的功能来维护系统是比较容易和轻松的。安装、卸载和升级 RPM 软件包只需一条命令就可以搞定。RPM 维护了一个所有已安装的软件包和文件的数据库，可以让用户进行查询和验证工作。在软件包升级过程中，RPM 会对配置文件进行特别处理，绝对不会丢失以往的定制信息。对于程序员 RPM 可以让我们连同软件的源代码打包成源代码和二进制软件包供最终用户使用。

RPM 拥有功能强大的查询选项。我们可以搜索数据库来查询软件包或文件。也可以查出某个文件属于哪个软件包或出自哪儿。RPM 软件包中的文件是以压缩格式存放的，拥有一个定制的二进制头文件，其中包含有关包和内容的信息，可以让我们对单个软件包的查询简便又快速。

RPM 另一个强大的功能是进行软件包的验证。如果我们担心误删了某个软件包中的某个文件，我们就可以对它进行验证。任何非正常现象将会被通知。如果需要的话还可以重新安装该软件包。在重新安装过程中，所有被修改过的配置文件将被保留。

RPM 设计目标之一就是要保持软件包的原始特征， 就象该软件的原始发布者发布软件时那样。通过使用 RPM 我们可以拥有最初的软件和最新的补丁程序，还有详细的软件构建信息。

概括的说：RPM 有五种基本的操作功能(不包括创建软件包)：安装、卸载、升级、查询、和验证。关于 rpm 命令的使用我们可以用以下命令：

PDF-XChange Click to buy NOW! www.docu-track.com

```
[root@hostname root]rpm -help
```

来获的。

2. RPM 工具功能

1） 安装

rpm -i (or --install) options file1.rpm ... fileN.rpm 通过 rpm -ivh 可以把 rpm 软件包安装到系统中，当然也可以使用不同的参数选项，笔者建议使用-ivh ，使用该选项可以解决大部分 rpm 软件包的安装，至于详细的参数说明可用查看 rpm 的 man 文档。

2 ）删除

rpm -e (or --erase) options pkg1 ... pkgN 如果某个软件包你再也不想使用了，那就用以上这个命令彻底的把你指定的 rpm 软件包清除掉把。

3 ）升级

rpm -U (or --upgrade) options file1.rpm ... fileN.rpm 由于开源软件更新速度快，用户当然要使用最新版本的软件包，此时最合适的就是 rpm 升级功能，当然最理想的参数选项就是-Uvh。

4 ）查询

rpm -q (or --query) options 实际上我们通常使用 rpm 工具最多的功能还是它的查询功能，比如查看软件包的版本、依赖关系等软件包的详细说明都要用到。最有用的参数选项是-qpi。

5 ）校验已安装的软件包

rpm -V (or --verify, or -y) options 一般我们可用通过该命令来验证已安装软件包，根据笔者的经验该命令一般没什么用途，只做一个了解就 ok 了。

3. spec 文件规范

能熟练掌握以上命令以及部分参数含义，管理日常的 rpm 软件包就不成问题了。然而随着 Linux 风靡全球，越来越多的开发者喜欢采用 RPM 格式来发布自己的软件包。那么 RPM 软件包是怎样制作的呢？对大多数 Linux 开发工程师来说是比较陌生的。

其实，制作 RPM 软件包并不是一件复杂的工作，其中的关键在于编写 SPEC

PDF-XChange
Click to buy NOW!
www.docu-track.com

软件包描述文件。要想制作一个 rpm 软件包就必须写一个软件包描述文件（SPEC）。这个文件中包含了软件包的诸多信息，如软件包的名字、版本、类别、说明摘要、创建时要执行什么指令、安装时要执行什么操作、以及软件包所要包含的文件列表等等。

描述文件说明如下：

（1） 文件头

一般的 spec 文件头包含以下几个域：

Summary：

用一句话概括该软件包尽量多的信息。

Name：

软件包的名字，最终 RPM 软件包是用该名字与版本号，释出号及体系号来命名软件包的。

Version：

软件版本号。仅当软件包比以前有较大改变时才增加版本号。

Release：

软件包释出号。一般我们对该软件包做了一些小的补丁的时候就应该把释出号加 1。

Vendor：

软件开发者的名字。

Copyright：

软件包所采用的版权规则。具体有：GPL（自由软件），BSD，MIT，Public Domain（公共域），Distributable（贡献），commercial（商业），Share（共享）等，一般的开发都写 GPL。

Group：

软件包所属类别，具体类别有：

Amusements/Games （娱乐/游戏）

PDF-XChange
Click to buy NOW!
www.docu-track.com

Amusements/Graphics（娱乐/图形）

Applications/Archiving （应用/文档）

Applications/Communications（应用/通讯）

Applications/Databases （应用/数据库）

Applications/Editors （应用/编辑器）

Applications/Emulators （应用/仿真器）

Applications/Engineering （应用/工程）

Applications/File （应用/文件）

Applications/Internet （应用/因特网）

Applications/Multimedia（应用/多媒体）

Applications/Productivity （应用/产品）

Applications/Publishing（应用/印刷）

Applications/System（应用/系统）

Applications/Text （应用/文本）

Development/Debuggers （开发/调试器）

Development/Languages （开发/语言）

Development/Libraries （开发/函数库）

Development/System （开发/系统）

Development/Tools （开发/工具）

Documentation （文档）

System Environment/Base（系统环境/基础）

System Environment/Daemons （系统环境/守护）

PDF-XChange
Click to buy NOW!
www.docu-track.com

System Environment/Kernel （系统环境/内核）

System Environment/Libraries （系统环境/函数库）

System Environment/Shells （系统环境/接口）

User Interface/Desktops（用户界面/桌面）

User Interface/X （用户界面/X 窗口）

User Interface/X Hardware Support （用户界面/X 硬件支持）

Source：

源程序软件包的名字。如 stardict-2.0.tar.gz。

%description：

软件包详细说明，可写在多个行上。

（2）%prep 段

这个段是预处理段，通常用来执行一些解开源程序包的命令，为下一步的编译安装作准备。%prep 和下面的%build，%install 段一样，除了可以执行 RPM 所定义的宏命令（以%开头）以外，还可以执行 SHELL 命令，命令可以有很多行，如我们常写的 tar 解包命令。

（3）build 段

本段是建立段，所要执行的命令为生成软件包服务，如 make 命令。

（4）%install 段

本段是安装段，其中的命令在安装软件包时将执行，如 make install 命令。

（5）%files 段

本段是文件段，用于定义软件包所包含的文件，分为三类--说明文档（doc），配置文件（config）及执行程序，还可定义文件存取权限，拥有者及组别。

（6）%changelog 段

本段是修改日志段。你可以将软件的每次修改记录到这里，保存到发布的软

PDF-XChange Click to buy NOW! www.docu-track.com

件包中，以便查询之用。每一个修改日志都有这样一种格式：第一行是：* 星期 月日 年 修改人电子信箱。其中：星期、月份均用英文形式的前 3 个字母，用中文会报错。接下来的行写的是修改了什么地方，可写多行。一般以减号开始，便于后续的查阅。

4. 打包

如果想发布 rpm 格式的源码包或者是二进制包，就要使用 rpmbuild 工具（rpm 最新打包工具）。如果我们已经根据本地源码包的成功编译安装而写了 spec 文件（该文件要以.spec 结束），那我们就可以建立一个打包环境，也就是目录树的建立，一般是在/usr/src/redhat/目录下建立 5 个目录。它门分别是 BUILD、SOURCE、SPEC、SRPM、RPM。其中 BUILD 目录用来存放打包过程中的源文件，SOURCE 用来存放打包是要用到的源文件和 patch，SPEC 用来存放 spec 文件，SRPM、RPM 分别存放打包生成的 rpm 格式的源文件和二进制文件。当然我们可以根据需要来选用不同的参数打包文件，笔者总结如下 3 条。

1） 只生成二进制格式的 rpm 包

```
rpmbuild -bb xxx.spec
```

用此命令生成软件包，执行后屏幕将显示如下信息：（每行开头为行号）

```
1 Executing:  %prep

2 + umask 022

3 + cd /usr/src/dist/BUILD

4 + exit 0

5 Executing:  %build

6 + umask 022

7 + cd /usr/src/dist/BUILD
```

生成的文件会在刚才建立的 RPM 目录下存在。

2）只生成 src 格式的 rpm 包

```
rpmbuild -bs xxx.spec
```

生成的文件会在刚才建立的 SRPM 目录下存在。

3） 只需要生成完整的源文件

```
rpmbuild -bp xxx.spec
```

源文件存在目录 BUILD 下。

读者朋友可能对这个命令不太明白，这个命令的作用就是把 tar 包解开然后把所有的补丁文件合并而生成一个完整的具最新功能的源文件。

4） 完全打包

```
rpmbuild -ba xxx.spec
```

产生以上 3 个过程分别生成的包。存放在相应的目录下。

软件包制作完成后可用 rpm 命令查询，看看效果。如果不满意的话可以再次修改软件包描述文件，重新运行以上命令产生新的 RPM 软件包。

二．典型 spec 文件分析

通过第一部分的介绍，我们对软件包的管理以及 spec 文件的一些细节应该掌握的差不多了，接下来通过分析 kaffeine.spec（kaffeine 是 linux 平台下的媒体播放器）文件来让读者朋友实践一回 spec 文件的规范和书写。

Kaffeine.spec 文件内容如下：

```
%define debug_package %

Name: kaffeine

Version: 0.4.3

Release: 25

Summary: A xine-based Media Player for KDE

Group: Applications/Multimedia

License: GPL

URL: http://kaffeine.sourceforge.net/

Source0: kaffeine-0.4.3.tar.bz2
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

```
Source1: logo.png

Source2: icon.tgz

Source3: kaffeine.desktop

Source4: codecs.tgz

Patch: kaffeine-0.4.3-fix-hide-crash.patch

Patch1:kaffeine-0.4.3-without-wizard.patch

BuildRoot: /var/tmp/kaffeine-root

%description

Kaffeine is a xine based media player for KDE3. It plays back CDs,

DVDs, and VCDs. It also decodes multimedia files like AVI, MOV, WMV,

and MP3 from local disk drives, and displays multimedia streamed over

the Internet. It interprets many of the most common multimedia formats

available - and some of the most uncommon formats, too. Additionally,

Kaffeine is fully integrated in KDE3, it supports Drag and Drop and

provides an editable playlist, a bookmark system, a Konqueror plugin,

a Mozilla plugin, OSD an much more.
```

以上这部分就是我们第一部分所说的文件头。这一部分主要包括软件包的名称、版本、源代码和 patch 等信息，通过这些关键字我们可以一目了然。查看以上内容，我们会全面了解该软件包。

接下来的这一个段就是核心部分，涉及到解包、补丁、编译、安装的过程。

```
%prep

%setup -q

%patch -p1
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

```
    %patch1 -p1

    %Build

    make -f admin/Makefile.common cvs

    ./configure --prefix=/usr

    make

    #for mo files

    pushd po

    rm *.gmo

    make

    popd

    %install

    mkdir -p $RPM_BUILD_ROOT

    make install DESTDIR=$RPM_BUILD_ROOT

    mkdir -p $RPM_BUILD_ROOT/usr/share/services

    cp $RPM_BUILD_ROOT/usr/share/apps/kaffeine/mms.protocol
$RPM_BUILD_ROOT/usr/share/services

    cp $RPM_BUILD_ROOT/usr/share/apps/kaffeine/rtsp.protocol
$RPM_BUILD_ROOT/usr/share/services

    #mkdir -p $RPM_BUILD_ROOT/usr/lib/firefox/plugins

    #cp $RPM_BUILD_ROOT/usr/lib/kaffeineplugin/kaffeineplugin.so

    $RPM_BUILD_ROOT/usr/lib/firefox/plugins

    cp % $RPM_BUILD_ROOT/usr/share/apps/kaffeine

    rm -rf $RPM_BUILD_ROOT/usr/share/icons/hicolor/*/apps/kaffeine.png
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

```
    rm -rf
$RPM_BUILD_ROOT/usr/share/icons/hicolor/*/apps/kaffeine-pause.png

    rm -rf
$RPM_BUILD_ROOT/usr/share/icons/hicolor/*/apps/kaffeine-play.png

    rm -rf
$RPM_BUILD_ROOT/usr/share/icons/hicolor/*/apps/kaffeine-record.png

    mkdir -p $RPM_BUILD_ROOT/usr/share/icons/crystalsvg

    tar zxvf % -C $RPM_BUILD_ROOT/usr/share/icons/crystalsvg

    mkdir -p $RPM_BUILD_ROOT/usr/share/applnk/App/Multimedia

    cp -r % $RPM_BUILD_ROOT/usr/share/applnk/App/Multimedia

    mkdir -p $RPM_BUILD_ROOT/usr/lib/win32

    tar zxvf % -C $RPM_BUILD_ROOT/usr/lib/win32

    %clean

    rm -rf $RPM_BUILD_ROOT

    %post

    ln -s /dev/cdrom /dev/dvd

    ln -s /dev/cdrom /dev/rdvd

    %files

    %defattr(-,root,root)

    /usr
```

这部分内容与所要打的包有关系，我们要根据具体情况来写出编译过程。这部分内容是最复杂的内容，当然，我们也可以看出，这样的写法其实就是在写一种规范化的脚本，说到脚本，读者朋友门就应该领会到这部分内容的灵活性了。

```
    %changelog

    * Fri Jul 1 2005 AiLin Yang -0.4.3-25
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

```
- modified the fullscreen bottom control panel

* Fri Jun 17 2005 xxx -0.4.3-24

- Modified to use xshm as video driver.

* Thu Jun 16 2005 AiLin Yang

- delete the option of Embed in system tray in configwidget

* Tue Jun 14 2005 AiLin Yang

- add fullscreen bottom control panel

- update kaffine to support my fullscreen bottom control panel
```

这部分内容可以说是 spec 文件的最后内容了，它对团队软件开发以及后续的软件维护至关重要，它相当于一个日志，记录了所有的基于该软件包的修改、更新信息。

小结

在 Linux 下 RPM 软件包的管理与 RPM 软件包的制作关键在 rpm 工具的使用和 spec 描述文件的起草。要想制作一个 RPM 格式的软件包必须编写软件包描述文件。其标准命名格式为:软件名-版本号-释出号.spec，这个文件详细描述了有关该软件包的诸多信息,如软件名,版本,类别,说明摘要,创建时要执行什么指令,安装时要执行什么操作,以及软件包所要包含的文件等等。有了这个文件 RPM 就可以制作出相应的 rpm 软件包。

附：中文 MAN

```
NAME

rpmbuild - 构建 RPM 打包

SYNOPSIS

构建打包：

rpmbuild [rpmbuild-options] SPECFILE ...

rpmbuild [rpmbuild-options] TARBALL ...
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

rpmbuild SOURCEPKG ...

其他：

rpmbuild --showrc

rpmbuild 选项

[--buildroot DIRECTORY] [--clean] [--nobuild]

[--rmsource] [--rmspec] [--short-circuit] [--sign]

[--target PLATFORM]

DESCRIPTION

rpmbuild 是用来构建软件的二进制和源代码打包的。一个软件包 package 包括文件的归档以及用来安装和卸载归档中文件的元数据。元数据包括辅助脚本，文件属性，以及有关的描述性的信息。软件包有两种 package：二进制软件包，用来封装要安装的软件，源代码软件包，包含了源代码和要构建二进制打包需要的内容。

必须选择下列基本模式之一：0 Build Package, Build Package from Tarball, Recompile Package, Show Configuration.

一般的选项

这些选项可以用于所有不同的模式。

-?, --help

输出较长的帮助信息

--version

输出一行信息，包含 rpmbuild 的版本号

--quiet

输出尽可能少的信息 - 通常只有错误信息才会显示出来

-v

输出冗余信息 - 通常常规的进度信息都将被显示

PDF-XChange Click to buy NOW! www.docu-track.com

-vv

输出大量丑陋的调试信息

--rcfile FILELIST

FILELIST 中冒号分隔的每个文件名都被 rpm 按顺序读取，从中获得配置信息。只有列表的第一个文件必须存在，波浪线将被替换为 $HOME。默认的 FILELIST 是 /usr/lib/rpm/rpmrc:/usr/lib/rpm/redhat/rpmrc:/etc/rpmrc:~/.rpmrc

--pipe CMD

将 rpm 的输出通过管道送到命令 CMD。

--dbpath DIRECTORY

使用 DIRECTORY 中的数据库，而不是默认的路径 /var/lib/rpm

--root DIRECTORY

以 DIRECTORY 作为根文件系统，进行所有操作。这意味着将使用 DIRECTORY 中的数据库来进行依赖性检测，任何小程序 (也就是安装中的 %post 和构建中的 %prep) 都将在一个 chroot(2) 到 DIRECTORY 之后执行。

构建选项

构建命令的一般形式是

rpmbuild -bSTAGE|-tSTAGE [rpmbuild-options] FILE ...

如果要用某个 spec 文件构建，使用 -b 参数。如果需要根据一个可能是压缩过的 tar 归档文件中的 spec 文件构建，就使用 -t 参数。第一个参数之后的字符 STAGE 指定了要完成的构建和打包的阶段，是下列其中之一：

-ba

构建二进制和源代码打包 (在执行 %prep, %build 和 %install 之后)

-bb

构建二进制打包 (在执行 %prep, %build 和 %install 之后)

PDF-XChange Click to buy NOW! www.docu-track.com

-bp

执行 spec 文件的 "%prep" 阶段。通常，这会解包源代码并应用补丁

-bc

执行 spec 文件的 "%build" 阶段 (在执行了 %prep 阶段之后)。这通常等价于执行了一次 "make"

-bi

执行 spec 文件的 "%install" 阶段 (在执行了 %prep 和 %build 阶段之后)。这通常等价于执行了一次 "make install"

-bl

执行一次 "列表检查"。spec 文件的 "%files" 段落中的宏被扩展，检测是否每个文件都存在。

-bs

只构建源代码打包

还可以用下列选项：

--buildroot DIRECTORY

在构建时，使用目录 DIRECTORY 覆盖默认的值

--clean

在制作打包之后删除构建树

--nobuild

不执行任何构建步骤。用于测试 spec 文件

--rmsource

在构建后删除源代码 (也可以单独使用，例如 "rpmbuild --rmsource foo.spec")

--rmspec

PDF-XChange
Click to buy NOW!
www.docu-track.com

在构建之后删除 spec 文件 (也可以单独使用，例如 "rpmbuild --rmspec foo.spec")

--short-circuit

直接跳到指定阶段 (也就是说，跳过指定阶段前面的所有步骤)。只有与 -bc 或 -bi 连用才有意义。

--sign

在打包中包含 GPG 签名。签名可以用来校验打包的完整性和来源。参见 rpm(8) 的 "GPG 签名" 章节中的配置细节。

--target PLATFORM

在构建时，将 PLATFORM 解析为 arch-vendor-os，并以此设置宏 %_target, %_target_cpu, 和 %_target_os 的值。

重建和重编译选项

还有两种发起构建的方法：

rpmbuild --rebuild|--recompile SOURCEPKG ...

这样执行的话，rpmbuild 安装指定的源代码打包，然后进行准备，编译和安装。另外，--rebuild 构建一个新的二进制打包，在构建结束时，构建目录被删除 (就好像用了 --clean)，源代码和 spec 文件也被删除。

SHOWRC

命令

rpmbuild --showrc

将显示 rpmbuild 使用的，在 rpmrc 和 macros 配置文件中定义的选项的值。

FILES

rpmrc 配置文件

/usr/lib/rpm/rpmrc

/usr/lib/rpm/redhat/rpmrc

PDF-XChange
Click to buy NOW!
www.docu-track.com

/etc/rpmrc

~/.rpmrc

Macro 宏定义文件

/usr/lib/rpm/macros

/usr/lib/rpm/redhat/macros

/etc/rpm/macros

~/.rpmmacros

Database 数据库

/var/lib/rpm/Basenames

/var/lib/rpm/Conflictname

/var/lib/rpm/Dirnames

/var/lib/rpm/Filemd5s

/var/lib/rpm/Group

/var/lib/rpm/Installtid

/var/lib/rpm/Name

/var/lib/rpm/Packages

/var/lib/rpm/Providename

/var/lib/rpm/Provideversion

/var/lib/rpm/Pubkeys

/var/lib/rpm/Removed

/var/lib/rpm/Requirename

/var/lib/rpm/Requireversion

PDF-XChange
Click to buy NOW!
www.docu-track.com

/var/lib/rpm/Sha1header

/var/lib/rpm/Sigmd5

/var/lib/rpm/Triggername

Temporary 临时文件

/var/tmp/rpm*