Creación de paquetes RPM (Red Hat Package Manager)

Anibal Jesús Avelar Rosales*

México 2003

Resumen

En muchas ocaciones necesitamos instalar algo sobre nuestra caja Linux, y lo primero que hacemos es hacer un script de instalación, en Shell o Perl, pero porque no hacer uso del sistema manejador de paquetes que trae el propio sistema. Este documento nos muestra la manera de construir nuestros propios paquetes. RPM es el gestor de paquetes de Red Hat (Red Hat Package Manager). Aunque aparece Red Hat en su nombre, es un sistema de empaquetado abierto y disponible para el uso de cualquiera. Permite a los usuarios tomar el código fuente (source code) y empaquetarlo en forma de fuentes y binaria de forma que los archivos binarios sean fácilmente instalables y rastreables y los fuentes puedan ser reconstruidas con facilidad. También gestiona una base de datos de todos los paquetes y sus archivos que puede ser usada para verificar paquetes e interrogarla para obtener información acerca de archivos y/o paquetes.

RPM es completamente flexible y fácil de usar, aunque provee la base para un sistema muy extenso. También es completamente abierto y disponible. Tal vez, el formato .deb de Debian sea mas poderoso, pero es un poco mas complejo a la hora de configurar y crear paquetes. Ademas, el uso de alguna distribución basada en Red Hat es mas difundido, así que aprender RPM es esencial.

Este documento describe el uso del formato de paquetes de instalación que se ha convertido en estándar de facto, el RPM (RedHat Package Manager).

*Del autor: Ingeniero en Computación egresado de Facultad de Ingeniería UNAM. Maestro en Ciencias de la Computación, IIMAS UNAM. Usuario Linux (fanático), administrador Unix(Solaris, HP, SGI, Linux), y programador Linux desde hace 6 años. Debian, Suse y Red Hat son mis favoritas, uso las 3. Alias: fixxxer. Yahoo: afixxxer, MSN: afixxxer@hotmail.com, ICQ: 98858676

Índice

1. Introducción

2. Información general

   2.1 Adquirir RPM

   2.2 Requerimientos de RPM

3. Instalar RPM

   3.1. BerkeleyDB y GCC

   3.2 Compilando RPM

4. Usando RPM

   4.1 Comandos básicos de RPM

   4.2 Comandos para construir RPM

   4.3 Comandos para hacer busquedas en RPM

   4.4 Comandos para verificar paquetes RPM

5. Algunos ejemplos útiles para probar RPM

6. Construyendo paquetes RPM

   6.1 El archivo rpmrc

   6.2 El archivo spec

   6.3 La Cabecera

   6.4 Las secciones %prep %build %install

   6.5 Guiones opcionales pre y post Install/Uninstall

   6.6 La sección %files

   6.7 Construcción

   6.8 Probándolo

   6.9 ¿Qué hacer con los nuevos paquetes RPM?

   6.10 ¿Y ahora qué?

7 Construcción multi-arquitectura de paquetes RPM

   7.1 Macros

   7.2 Excluyendo arquitectura de los paquetes

8. Usando paquetes Debian - Convertiendo archivos .deb archivos .rpm

9. Comparativa entre RPM y DEB

1. Introducción

RPM es el gestor de paquetes de Red Hat (Red Hat Package Manager). Aunque aparece Red Hat en su nombre, la intención es que sea un sistema de empaquetado abierto y disponible para el uso de cualquiera. Permite a los usuarios tomar el código fuente (source code) y empaquetarlo en forma de fuentes y binarios de forma que los archivos binarios sean fácilmente instalables y rastreables y los archivos fuentes puedan ser reconstruidos con facilidad. También gestiona una base de datos de todos los paquetes y sus archivos que puede ser usada para verificar paquetes o interrogarla para obtener información acerca de archivos y/o paquetes.

Red Hat Software anima a otros vendedores de distribuciones a dedicar un rato para examinar RPM y usarlo para sus propias distribuciones. RPM es completamente flexible y fácil de usar, aunque provee la base para un sistema muy extenso. También es completamente abierto y disponible. Se concede permiso para usar y distribuir RPM, bajo la protección de la licencia GP.

Con RPM, si sale disponible una nueva versión de un programa, usted no necesita empezar desde la nada para conseguir que compile. Puede examinar el parche para saber qué podría necesitar hacer. De esta manera toda la configuración por defecto de compilación queda fácilmente a la vista.

RPM también está diseñado para disponer de potentes parámetros de consulta. Usted puede hacer búsquedas de paquetes a lo largo de toda la base de datos o sólo de ciertos archivos. También puede encontrar fácilmente a qué paquete pertenece un archivo y de dónde proviene. Los archivos RPM en sí mismos son archivos comprimidos, pero puede consultar paquetes independientes fácil y rápidamente, gracias a una cabecera binaria a medida añadida al paquete con toda la información que puede necesitar, almacenada sin comprimir. Esto permite consultas rápidas.

Otra poderosa característica es la habilidad de verificar paquetes. Si está preocupado por haber borrado algún archivo importante, sólo tiene que verificar el paquete. Quedará cumplidamente informado de cualquier anomalía. Llegados a ese punto, podrá reinstalar el paquete si lo considera necesario. Cualquier archivo de configuración que usted tenga quedará a salvo.

2. Información general

2.1 Adquirir RPM

La mejor forma de conseguir RPM es instalando Red Hat Commercial Linux. Si no quiere hacer eso, puede seguir usando el fuente de RPM. Puede conseguirse en:

El sitio oficial de RPM es http://www.rpm.org, y los fuentes están disponibles en:

ftp://ftp.rpm.org/pub/rpm o bien en desde Red Hat

ftp://ftp.redhat.com/pub/redhat/linux/code/rpm.

Puede encontrar información más completa sobre RPM en el libro de Ed Bailey Maximum RPM. Dicho libro está disponible en www.redhat.com. O desde mi sitio en Internet en http://www.cofradia.org/~aavelar/html/consol2003

2.2 Requerimientos de RPM

El principal requerimiento para ejecutar RPM es cpio 2.4.2 o superior. Aunque el sistema fue ideado para ser usado con Linux, puede ser perfectamente portado a cualquier sistema Unix. De hecho, ha sido compilado en SunOS, Solaris, AIX, Irix, Sinix, NetBSD, UnixWare, AmigaOS, SCO, y otros. Queda advertido que los paquetes binarios generados en diferentes tipos de sistemas Unix no serán compatibles entre sí.

Estos son los mínimos requerimientos para instalar RPMs. Para construir RPMs a partir de los fuentes, necesitará todo lo normalmente requerido para construir un paquete, cosas como gcc, make, etc.

3. Instalar RPM

3.1 BerkeleyDB y GCC

Para instalar los fuentes de RPM usted necesita el GCC y BerkeleyDB. Primero usted debe decidir la versión de Redhat Linux que usted desea utilizar(sí lo hace sobre Linux y sobre RedHat). OtrosLinux no deben tener problemas como Mandrake, Suse, Debian etc.

3.2 Compilando RPM

Para construir el paquete de la RPM usted necesita el software de BerkeleyDB y el compilador GCC. El compilador GCC ya existe casi para todas las plataformas, así que no será problema. Es muy importante que usted debe seleccionar versiones apropiadas del GCC, de BerkeleyDB y de la fuente de la RPM según sus propias necesidades. Lo que hay que tener cuidado es que con viejas versiones de fuentes RPM o de BerkeleyDB los paquetesRPM que se construyan no serán compatibles con las mas nuevas versiones. Así que si hacemos un sistema de RPM que queremos sea compatible entre sí para diferentes plataformas tenemos que usar versiones compatibles. Los fuentes de BerkeleyDB se pueden obtener del sitio http://www.sleepycat.com

4. Usando RPM

4.1 Comandos básicos de RPM

bash# rpm -ivh myrpm-1.0-1.i386.rpm ...(para instalar paquetes)

bash# rpm -ivh ftp://ftp.redhat.com/pub/redhat/RPMS/myrpm-1.0-1.i386.rpm ...(para instalar paquetes vía ftp)

bash# rpm -e myrpm ...(para desinstalar paquetes)

bash# rpm -U myrpm-1.0-1.i386.rpm ...(para actualizar un paquete a una versión mas actual)

bash# rpm –help ...(para ver todas las opciones del comando rpm)

En su forma más simple, RPM puede usarse para instalar paquetes:

Uno de las cosas mas útiles permiten instalar paquetes a través de FTP. Si está conectado a la Red y quiere instalar un nuevo paquete, todo lo que necesita hacer es especificar el archivo con un URL válido, como esto:

bash# rpm -i ftp://ftp.suse.com/pub/linux/8.0/RPMS/lyx-1.0-1.i386.rpm

Con esto se ve que con RPM puede hacer consultas y/o instalaciones a través de FTP. Aunque estos son comandos simples, rpm puede usarse de multitud de formas, como puede verse en el mensaje de Ayuda:

bash$ rpm –help

RPM version 3.0

Copyright (C) 1998 - Red Hat Software

This may be freely redistributed under the terms of the GNU Public License

usage: rpm {–help}

rpm {–version}

rpm {–initdb} [–dbpath <dir>]

rpm {–install -i} [-v] [–hash -h] [–percent] [–force] [–test]
[–replacepkgs] [–replacefiles] [–root <dir>]
[–excludedocs] [–includedocs] [–noscripts]
[–rcfile <file>] [–ignorearch] [–dbpath <dir>]
[–prefix <dir>] [–ignoreos] [–nodeps]
[–ftpproxy <host>] [–ftpport <port>]
file1.rpm ... fileN.rpm

rpm {–upgrade -U} [-v] [–hash -h] [–percent] [–force] [–test]
[–oldpackage] [–root <dir>] [–noscripts]
[–excludedocs] [–includedocs] [–rcfile <file>]
[–ignorearch] [–dbpath <dir>] [–prefix <dir>]
[–ftpproxy <host>] [–ftpport <port>]
[–ignoreos] [–nodeps] file1.rpm ... fileN.rpm

rpm {–query -q} [-afpg] [-i] [-l] [-s] [-d] [-c] [-v] [-R]
[–scripts] [–root <dir>] [–rcfile <file>]
[–whatprovides] [–whatrequires] [–requires]
[–ftpuseport] [–ftpproxy <host>] [–ftpport <port>]
[–provides] [–dump] [–dbpath <dir>] [targets]

rpm {–verify -V -y} [-afpg] [–root <dir>] [–rcfile <file>]

[–dbpath <dir>] [–nodeps] [–nofiles] [–noscripts]
[–nomd5] [targets]
rpm {–setperms} [-afpg] [target]
rpm {–setugids} [-afpg] [target]
rpm {–erase -e} [–root <dir>] [–noscripts] [–rcfile <file>]
[–dbpath <dir>] [–nodeps] [–allmatches]
package1 ... packageN
rpm {-b|t}[plciba] [-v] [–short-circuit] [–clean] [–rcfile <file>]
[–sign] [–test] [–timecheck <s>] specfile
rpm {–rebuild} [–rcfile <file>] [-v] source1.rpm ... sourceN.rpm
rpm {–recompile} [–rcfile <file>] [-v] source1.rpm ... sourceN.rpm
rpm {–resign} [–rcfile <file>] package1 package2 ... packageN
rpm {–addsign} [–rcfile <file>] package1 package2 ... packageN
rpm {–checksig -K} [–nopgp] [–nomd5] [–rcfile <file>]
package1 ... packageN
rpm {–rebuilddb} [–rcfile <file>] [–dbpath <dir>]
rpm {–querytags}

Esta es la versión de ayuda de la versión 3.0, que compile para Suse 8.0 y funciona de maravilla. Podrá encontrar más detalles acerca de la función de estos parametros en la página del manual de RPM.

4.2 Comandos para construir RPM

bash# rpm -i myrpm*.src.rpm

bash# cd /usr/src/redhat/SPECS

bash# rpm -ba myrpm-1.0-1.spec (para construir el paquete de manera completa)

Para construir un paquete en pasos incremental, hacer:

bash# rpm -bp myrpm-1.0-1.spec ...(para ejecutar la seccion %prep)

bash# rpm –short-circuit -bc myrpm-1.0-1.spec ...(para ejecutar solo la sección %build)

bash# rpm –short-circuit -bi myrpm-1.0-1.spec ...(para ejecutar solo la sección %install)

bash# rpm -ba myrpm-1.0-1.spec ... (para construir el paquete de manera completa)

4.3 Comandos para hacer busquedas en RPM

bash$ rpm -qpl myrpm-1.0-1.i386.rpm ....(lista de archivos en un paquete RPM no instalado)

bash$ rpm -ql myrpm-1.0-1 ....(lista de archivos de un paquete instalado)

bash$ rpm -qpR myrpm-1.0-1.i386.rpm ....(lista de paquetes de cual el paquete aun no instalado depende)

bash$ rpm -qR myrpm-1.0-1 ....(lista de paquetes de cual el paquete instalado depende)

bash$ rpm -q myrpm ...(imprime el nombre del paquete, versión, etc.)

bash$ rpm -qa | less ....(lista todos los paquetes instalados)

bash$ rpm -qa | grep -i kde ....(lista todos los instalados paquetes que conciden con kde)

bash$ rpm -qif /bin/ls ....(lista el paquete en el cual esta instalado el archivo /bin/ls)

Para ver todas las opciones que se pueden establecer un los archivos rpmrc(/usr/lib/rpm/rpmrc, /etc/rpmrc, ~/.rpmrc):

bash$ rpm –showrc | less

5. Algunos ejemplos útiles para probar RPM

RPM es una herramienta potentísima y, como puede ver, dispone de varios parámetros. La mejor forma de ver como funcionan es examinando unos cuantos ejemplos. Ahora mostraremos unos ejemplos muy útiles y que nos servirán en cualquier momento:

- Supongamos que ha borrado unos cuantos archivos por accidente, pero no está seguro de qué es lo que ha borrado. Si quiere verificar completamente su sistema y ver qué se ha perdido, puede hacer:

  bash$ rpm -Va

  - O bien verificar si falta un archivo a un solo paquete que ya esta instalado.

  bash$ rpm -Vl myrpm-1.0-0

  missing /opt/myrpm/maximum-rpm.ps

  Nos dice que hace falta el archivo maximum-rpm.ps en nuestro paquete. Debemos reinstalarlo, hay dos formas:

  bash$ rpm -e myrpm-1.0-0 (Lo borramos)

  bash$ rpm -ivh myrpm-1.0-0.i386.rpm (lo instalos de nuevo)

  Sin embargo, mas facilmente se puede hacer esto:

  bash$ rpm -ivh –replacepkgs myrpm-1.0-0.i386.rpm (resinstala todo remplazando archivos).

  bash$ rpm -ivh –force myrpm-1.0-0.i386.rpm (reinstala todo forzando).

- Supongamos que se encuentra con un archivo que no reconoce. Para saber a qué paquete pertenece puede hacer:

bash$ rpm -qf /sbin/shutdown

La salida podría ser:

sysvinit-2.82-9

▷ Supongamos que tiene un paquete RPM, pero no sabe qué puede ser. Para obtener información al respecto:

bash$ rpm -qpi minicom-2.00.0-70.i386.rpm

La salida podría ser:

Name : minicom Relocations: (not relocateable)

Version : 2.00.0 Vendor: SuSE AG, Nuernberg, Germany

Release : 70 Build Date: vie 12 abr 2002 09:29:38 CST

Install date: (not installed) Build Host: Johnson.suse.de

Group : Hardware/Modem Source RPM: minicom-2.00.0-70.src.rpm

Size : 685047 License: GPL

Packager : feedback@suse.de

Summary : A terminal program

Description :

Terminal program similar to Telix(tm) (program for calling other computers via modem) under MS-DOS.

If you want to access your Modem with minicom, you have to be in the group uucp.

Authors: ——– Miquel van Smoorenburg <miquels@drinkel.ow.nl>

SuSE series: n Distribution: SuSE Linux 8.0 (i386)

▷ Ahora quiere saber qué archivos instala el paquete RPM. Puede hacer:

bash$ rpm -qpl minicom-2.00.0-70.i386.rpm

La salida es:

/usr/bin/ascii-xfr

/usr/bin/minicom

/usr/bin/runscript

/usr/bin/xminicom

/usr/share/doc/packages/minicom

/usr/share/doc/packages/minicom/ABOUT-NLS

/usr/share/doc/packages/minicom/AUTHORS
/usr/share/doc/packages/minicom/Announce-1.78
/usr/share/doc/packages/minicom/Announce-1.83
/usr/share/doc/packages/minicom/COMPATABILITY.lrzsz
/usr/share/doc/packages/minicom/COPYING
/usr/share/doc/packages/minicom/ChangeLog
/usr/share/doc/packages/minicom/ChangeLog.old
/usr/share/doc/packages/minicom/FILE_ID.DIZ
/usr/share/doc/packages/minicom/HistSearch
/usr/share/doc/packages/minicom/INSTALL
/usr/share/doc/packages/minicom/Locales
/usr/share/doc/packages/minicom/Macros
/usr/share/doc/packages/minicom/Makefile
/usr/share/doc/packages/minicom/Makefile.am
/usr/share/doc/packages/minicom/Makefile.in
/usr/share/doc/packages/minicom/NEWS
/usr/share/doc/packages/minicom/QuickStart.modemu
/usr/share/doc/packages/minicom/README
/usr/share/doc/packages/minicom/README.lrzsz
/usr/share/doc/packages/minicom/RedHat
/usr/share/doc/packages/minicom/TODO.lrzsz
/usr/share/doc/packages/minicom/Todo
/usr/share/doc/packages/minicom/Todo.175
/usr/share/doc/packages/minicom/Todo.Irix.dif
/usr/share/doc/packages/minicom/Todo.emacskey.dif
/usr/share/doc/packages/minicom/Todo.fsel
/usr/share/doc/packages/minicom/copyright.modemu
/usr/share/doc/packages/minicom/fselector.txt
/usr/share/doc/packages/minicom/japanese
/usr/share/doc/packages/minicom/minicom.FAQ
/usr/share/doc/packages/minicom/minicom.users
/usr/share/doc/packages/minicom/minirc.dfl
/usr/share/doc/packages/minicom/modemu.README
/usr/share/doc/packages/minicom/pl-translation.txt

/usr/share/doc/packages/minicom/portugues-brasil

/usr/share/doc/packages/minicom/suomeksi

/usr/share/locale/cs_CZ/LC_MESSAGES/minicom.mo

/usr/share/locale/es/LC_MESSAGES/minicom.mo

/usr/share/locale/fi_FI/LC_MESSAGES/minicom.mo

/usr/share/locale/fr/LC_MESSAGES/minicom.mo

/usr/share/locale/ja/LC_MESSAGES/minicom.mo

/usr/share/locale/ja_JP.SJIS/LC_MESSAGES/minicom.mo

/usr/share/locale/pl/LC_MESSAGES/minicom.mo

/usr/share/locale/pt_BR/LC_MESSAGES/minicom.mo

/usr/share/locale/ru/LC_MESSAGES/minicom.mo

/usr/share/man/man1/ascii-xfr.1.gz

/usr/share/man/man1/minicom.1.gz

/usr/share/man/man1/runscript.1.gz

Estos son sólo unos pocos ejemplos. Otros, aún más creativos, podrá hacerlos fácilmente una vez que se haya familiarizado con RPM.

6. Construyendo paquetes RPM

Construir paquetes RPM es algo realmente fácil de hacer, especialmente si puede conseguir que el software que intenta empaquetar pueda compilarse por sí mismo.

El procedimiento básico es el siguiente:

- Asegúrese que su /etc/rpmrc está configurado para su sistema.
- Hágase con el código fuente del software a empaquetar para ser compilado en su sistema. Casi siempre en un archivo *.tar.gz
- Haga un parche con todos los cambios que ha tenido que realizar para que los fuentes se compilen adecuadamente. Esto es necesario si tuvo que modificar opciones de compilación o rutas.
- Cree un fichero .spec para el paquete.
- Asegúrese de que todo está en su sitio.
- Construya el paquete usando RPM.

Por default, RPM construirá tanto el paquete binario como el de los fuentes.

6.1 El archivo rpmrc

La única configuración de RPM está disponible en el archivo rpmrc. Este varía su ubicación según la distribucion de linux o sistemas operativo. En mi versión 3.0 que compile sobtre Suse 8.0 quedo en /usr/local/lib/rpmrc. Éste puede tener la siguiente apariencia:

```
optflags: i386 -O2 -m486 -fno-strength-reduce
optflags: alpha -O2
optflags: sparc -O2
optflags: sparc64 -O2
optflags: m68k -O2 -fomit-frame-pointer
optflags: ppc -O2 -fsigned-char
optflags: parisc -O2 -mpa-risc-1-0
optflags: hppa1.0 -O2 -mpa-risc-1-0
...
##############################################################################
# Canonical arch names and numbers
arch_canon: i986: i986 1
arch_canon: i886: i886 1
arch_canon: i786: i786 1
arch_canon: i686: i686 1
arch_canon: i586: i586 1
arch_canon: i486: i486 1
arch_canon: i386: i386 1
arch_canon: alpha: alpha 2
arch_canon: sparc: sparc 3
arch_canon: sun4: sparc 3
arch_canon: sun4m: sparc 3
arch_canon: sun4c: sparc 3
arch_canon: sun4d: sparc 3
...
##############################################################################
# Canonical OS names and numbers
os_canon: Linux: Linux 1
os_canon: IRIX: Irix 2 # This is wrong
os_canon: SunOS5: solaris 3
os_canon: SunOS4: SunOS 4
os_canon: AmigaOS: AmigaOS 5
os_canon: AIX: AIX 5
os_canon: HP-UX: hpux10 6
os_canon: OSF1: osf1 7
os_canon: osf4.0: osf1 7
os_canon: osf3.2: osf1 7
```

os_canon: FreeBSD: FreeBSD 8
os_canon: SCO_SV: SCO_SV3.2v5.0.2 9
...
macrofiles:/usr/local/rpm/lib/rpm/macros:/usr/local/rpm/lib/rpm/%{_target}/macros:/etc/rpm/mac
#include: /usr/local/rpm/lib/rpm/%{_target}/rpmrc

RPM también soporta ahora el empaquetado de paquetes para múltiples arquitecturas. El archivo rpmrc puede incluir una variable de opciones (optflags) que contiene parámetros específicos a la arquitectura. Lea secciones posteriores para saber cómo usar esta variable.

En adición a las macros anteriores, hay disponibles unas cuantas más. Puede usar:

bash$ rpm –showrc

ARCHITECTURE AND OS:

build arch : i386 compatible

build archs: i686 i586 i486 i386 noarch

build os : Linux compatible

build os's : Linux

install arch : i686

install os : Linux

compatible archs : i686 i586 i486 i386 noarch

compatible os's : Linux

RPMRC VALUES:

macrofiles : /usr/lib/rpm/macros:/usr/lib/rpm/i686-linux/macros:/usr/lib/rpm/suse_macros:/etc/rp
linux/ macros:~/.rpmmacros

optflags : -O2 -march=i686 -mcpu=i686

...

para saber cuál de sus etiquetas están configuradas y cuáles son los parámetros disponibles.

6.2 El archivo spec

Este archivo es imprescindible para construir un paquete. El archivo spec es una descripción del software acompañado con instrucciones sobre cómo construirlo y una lista de archivos de todos los binarios instalados.

Debería nombrar a sus archivos spec de acuerdo a una convención estándar. Tal como nombre de paquete-guión-número de versión-guión-número de publicación-punto-spec.

A continuación, un pequeño archivo spec. El paquete es mi pquete de este tutorial que llame myrpm-1.0 y el archivo spec es myrpm-1.0.spec.

El contenido es el siguiente:

Mi paquete

%define name myrpm

%define version 1.0

%define release 1

%define serial 1

%define prefix /opt/myrpm

Summary: Test

Name: myrpm

Version: 1.0

Release: 0

Copyright: GPL

Vendor: Fixxxer Inc.

Group: Development/Tools/Test

Source: %{name}-%{version}.tar.gz

URL: http://www.cofradia.org/~aavelar

Packager: Anibal J. Avelar <aavelar@cofradia.org>

BuildRoot: /var/tmp/ %{name}-%{version}

%description

Este es el tutorial de de 'Creacion de RPM (Red Hat Package Manager)'. Trae el manual completo de RPM de Red Hat y los archivos PDF y PS del tutorial d el consol 2003.

Esta parte la ejecuta antes de ejecutar el cualquier cosa

%pre rm -rf %{prefix}

Esta parte la ejecuta inmediatamente despues de que instalo el paquete RPM %post

%prep

%setup

%build

Nada que hacer ...

Instalamos archivo maximum-rpm. Y los archivos del PDF y PS del tutorial.

%install

rm -rf $RPM_BUILD_ROOT

mkdir -p $RPM_BUILD_ROOT%{prefix}

install -c *.ps $RPM_BUILD_ROOT%{prefix}

install -c *.pdf $RPM_BUILD_ROOT%{prefix}

Ahora elaboramos la lista de archivos

cd $RPM_BUILD_ROOT%{prefix}

###Esto se ejecuta antes de desinstalar un paquete

%preun

###Esto se ejecuta exactamente despues de que se desinstalo el paquete

%postun rm -rf %{prefix}

%clean #echo rm -rf $RPM_BUILD_ROOT $RPM_BUILD_ROOT/file.list. %{name}

##Esta parte es la que define que archivos van en el paquete de RPM

##Los nombres de subdirectorios pueden ser del sistema. Aqui habra que tener cui dado ya que sobre-escribe permisos

%files

%defattr(-,root,root)

%{prefix}/*

##Se instalan todos los archivos(*) en el directorio {prefix}. Se pudo ir instalando archivo por archivo

de esta forma /opt/myrpm/consol2003-RPM.ps

6.3 La Cabecera

La cabecera del archivo SPEC tiene unos cuantos campos estándar que usted necesita rellenar. También hay unas cuantas advertencias. Los campos deben ser rellenados tal como sigue:

* Summary: Descripción en una sóla línea del paquete.

* Name: La cadena que vaya a servir de nombre para el archivo rpm.

* Version: La cadena que vaya a ser el número de versión para el archivo rpm.

* Release: El número de publicación para un paquete dentro de un mismo número de versión (ej.: si crea un paquete y lo encuentra ligeramente defectuoso y necesita generarlo de nuevo, el siguiente paquete debería tener 2 como número de publicación).

* Icon: El nombre del icono que podrán usar interfaces de instalación alto nivel (como Red Hat 'glint"). Debe ser un archivo .gif y estar situado en el directorio SOURCES.

* Source: Esta línea apunta a la localización HOME del archivo de fuentes original. Se usa si alguna vez quiere tener los fuentes de nuevo o chequear para nuevas versiones. Precaución: el nombre del archivo en esta línea DEBE coincidir con el nombre que tiene tal archivo en su sistema (ej.: no se haga con el archivo fuente y le cambie el nombre). Puede especificar más de un archivo fuente de esta forma:

  Source0: blah-0.tar.gz

  Source1: blah-1.tar.gz

Source2: fooblah.tar.gz

Estos archivos deben residir en el directorio SOURCES. (La estructura de directorios es discutida en una sección posterior, "El Árbol de Directorios de las Fuentes", arbol).

* Patch: El lugar donde podrán encontrarse los parches si los necesita de nuevo. Precaución: el nombre debe coincidir con el de SUS propios parches. Puede especificar más de un archivo de parches de esta forma:

  Patch0: blah-0.patch

  Patch1: blah-1.patch

  Patch2: fooblah.patch

Estos archivos deben residir en el directorio SOURCES.

* Copyright: Hace referencia al modelo de copyright al que se acoje el paquete. Puede tratarse de algo al estilo de GPL, BSD, MIT, etc.

* BuildRoot: Hace referencia a un directorio que simulará el raíz (/) para la construcción e instalación de un nuevo paquete. Puede usarlo para probar su paquete antes de instalarlo en su máquina.

* Group: Informa a un programa de instalación de alto nivel (como Red Hat 'glint") dónde situar este paquete en particular dentro de la jerarquía de rpm. Esta jerarquía podría ser:

  Diversión

  Desarrollo

  Documentación

  Hardware

  Productividad

  Sistema

%description

En realidad no es un elemento de la cabecera, pero debe ser descrito junto a sus otras partes. Necesita una etiqueta de descripción por cada paquete o subpaquete. Se trata de un campo multilínea que debe ser usado para proporcionar una descripción comprensible del paquete.

Estas categorías a su vez se dividen en subcategorías, y estas en otras subcategorías. Lo ideal a la hora de de elegir un grupo es escojer un grupo que exista en el sistema, para que las herramientas de control de paquetes peudan ver nuestro recien instalado paquete.

6.4 Las secciones %prep %build %install

%prep: Esta es la segunda sección del archivo spec. Se usa para tener las fuentas listas para compilar. Aquí necesita hacer todo lo necesario para obtener los fuentes parcheadas y configuradas para ejecutar un make con ellas. Cada una de estas secciones es sólo un lugar para ejecutar guiones de intérprete de comandos. Así podrá crear un script simple para sh y colocarlo tras la etiqueta %prep para desempaquetar y parchear sus fuentes. En cualquier caso, hemos creado unas macros para ayudar en esto. La primera de estas macros es %setup. En su forma más simple (sin parámetros en la línea de comandos), se limita a desempaquetar los fuentes y cambiar el directorio actual al de los fuentes. Puede tener alguna de las siguientes opciones:

-n nombre asignará el nombre del directorio en construcción. Por defecto es $NAME-$VERSIÓN. Otras posibilidades incluyen $NAME, ${NAME}${VERSIÓN}, o cualquiera que use el archivo tar principal. (Cheque que estas variables "$" no son variables reales dentro del archivo spec. Sólo se usan aquí en lugar de un nombre ejemplo. Necesitará usar el nombre real y la versión de su paquete, no una variable).

-c creará y cambiará al directorio nombrado antes de desempaquetar con tar.

-b # desempaquetará con tar el archivo fuente # antes de cambiar al directorio (y esto no tiene sentido con -c, así que no lo haga). Sólo es útil cuando se usan varios archivos fuente.

-a # desempaquetará el archivo fuente # después de cambiar al directorio.

-T Este parámetro anula la acción por defecto de desempaquetar el archivo fuente y requiere -b 0 o -a 0 para desempaquetar el archivo fuente principal. Necesitará esto cuando hayan fuentes secundarias.

-D No borra el directorio antes de desempaquetar. Sólo resulta útil cuando tenga más de una macro de configuración. Debería ser usado solamente en macros de configuración después de la primera (pero nunca en la primera).

La siguiente de las macros disponibles es %patch. Esta macro ayuda a automatizar el proceso de aplicación de parches a los fuentes. Necesita de varios parámetros, listadas a continuación:

- # aplicará el parche Patch#
- -p# especifica el número de directorios a evitar por el comando patch(1).

- -P La acción por defecto es aplicar Patch (o Patch0). Este parámetro inhibe dicha acción y requiere un 0 para tener desempaquetado el archivo fuente principal. Esta opción resulta útil en una segunda (o posterior) macro %patch que requiera parámetros distintos a la primera macro.
- También puede hacer %patch# en lugar de hacer el comando real: %patch # -P

Estas deberían ser todas las macros que necesite. En cuanto las tenga claras, podrá crear cualquier otra configuración que necesite mediante guiones sh. Todo lo que incluya hasta la macro %build (discutida en la siguiente sección) es ejecutado vía sh. Busque en el ejemplo anterior el tipo de cosas que puede querer incluir allí.

%build: En realidad no hay ninguna macro en esta sección. Solamente debe incluir todos los comandos que necesitaría para construir y/o compilar el software una vez que haya desempaquetado y parcheado los fuentes, y se haya movido al directorio correcto. Es pues otra serie de comandos pasados a sh, así que cualquier comando aceptable por sh podrá ir aquí (incluidos los comentarios). El directorio actual es reajustado en cada una de estas secciones al de mayor nivel en el directorio de fuentes, así que téngalo en cuenta. Puede moverse a través de los subdirectorios si resultase necesario.

%install: En realidad no hay ninguna macro en esta sección. Básicamente debe incluir aquí cualquier comando necesario para instalar. Si el paquete a construir tiene disponible un comando make install, inclúyalo aquí. Si no, o bien puede parchear el archivo Makefile y añadirle la funcionalidad make install e incluir dicha sentencia en esta sección o bien instalarlo a mano mediante comandos sh. Puede considerar su directorio actual como el directorio superior del directorio de fuentes.

6.5 Guiones opcionales pre y post Install/Uninstall

Puede incluir guiones que serían ejecutados antes y después de la instalación o desinstalación de paquetes binarios. Una razón importante para esto es hacer cosas como ejecutar ldconfig tras la instalación o eliminar paquetes que contienen librerías compartidas. O borrar subdirectorios que no fueron borrados al desinstalar. O algún proceso que requiera ser ejecutado antes/despues de des/instalar. Las macros para cada uno de los guiones son:

- %pre es la macro para los guiones pre-instalación.
- %post es la macro para los guiones post-instalación.
- %preun es la macro para los guiones pre-desinstalación.
- %postun es la macro para los guiones post-desinstalación.

Los contenidos de estas secciones deben ser solamente guiones sh, luego no resulta necesaria la línea #!/bin/sh.

6.6 La sección %files

Esta es la sección donde debe listar los archivos del paquete binario. RPM no tiene forma de saber qué archivos binarios se han instalado tras ejecutar make install. NO hay forma de saberlo.

Algunos han sugerido ejecutar un comando find antes y después de la instalación del paquete. En un sistema multiusuario, esto es inaceptable ya que pueden crearse otros archivos durante la construcción del paquete que no tienen nada que ver con el mismo.

También hay algunas macros disponibles para hacer cosas especiales. Son las listadas a continuación:

- %doc se usa para señalar los archivos de documentación del paquete fuente que desea que sean instalados en una instalación binaria. La documentación será instalada en /usr/doc/$NOMBRE-$VERSIÓN-$PUBLICACIÓN. La lista podrá incluir varios archivos en una sóla línea o puede listarlos de forma separada con una macro para cada uno de ellos.
- %config se usa para señalar los archivos de configuración en un paquete. Archivos así pueden ser sendmail.cf, passwd, etc. Si posteriormente desinstala un paquete que incluye archivos de configuración, todos los archivos sin modificar serán borrados y todos los archivos modificados serán movidos a su nombre antiguo con el sufijo .rpmsave añadido a su nombre. También puede incluir múltiples archivos con esta macro.
- %dir señala a un único directorio en la lista como propiedad de un paquete. Por defecto, si incluye en la lista un nombre de directorio SIN una macro %dir, TODO el contenido de ese directorio es incluido en la lista de archivos y posteriormente instalado como parte del paquete.
- %files -f <nombrede_archivo>le permitirá tener la lista de archivos contenida en un archivo situado en el directorio de las fuentes. Resulta útil en los casos en los que un paquete puede crear su propia lista de archivos por sí mismo. En ese caso sólo tendrá que incluir el nombre de ese archivo aquí y no necesitará especificar nada más.

La mayor precacución que debe tener en cuenta en la lista de archivos es la inclusión de directorios. Si por accidente incluye /usr/bin, su paquete binario contendrá todos los archivos contenidos en el directorio /usr/bin en su sistema. Es decir, podríamos sobre-escribirlo accidentalmente.

6.7 Construcción

Lo primero que necesita es un árbol de directorios de compilación configurado de forma apropiada. Esto se puede hacer mediante el archivo rpmrc. La mayoría

de la gente sólo usará /usr/src. El sistema RPM que compile a mano lo dejo en /usr/local/rpm/src.

Puede que necesite crear los siguientes directorios para organizar un árbol de construcción, si no es que a la hora de compilar (si lo hicimos a mano) o de instalar RPM se crearon automáticamente:

- BUILD es el directorio donde RPM lleva a cabo toda la construcción. No tiene que llevar a cabo su prueba de construcción en ningún sitio en particular; aquí es donde RPM llevará a cabo la compilación y empaquetamiento.
- SOURCES es el directorio donde debe situar los archivos fuente originales y los correspondientes parches. Es donde RPM buscará por defecto.
- SPECS es el directorio donde deben ir situados los archivos spec. En este se pondrá el archivo spec si instalamos un paquete de fuentes. Por ejemplo: myrpm-1.0-0.src.rpm, instala myrpm-1.0.spec en este directorio.
- RPMS es donde RPM dejará los paquetes RPM binarios una vez construidos. Se divide por arquitectura por default es i386. Pero si se compilara con la opción –buildarch arch, se generaría en el directorio correspondiente.
- SRPMS es donde RPM dejará los paquetes RPM fuentes.

Prueba de construcción de un paquete RPM

Lo primero que querrá hacer es asegurarse que los fuentes se construyen adecuadamente sin usar RPM. Para ello, desempaquete los fuentes. Cree un subdirectorio llamado por ejemplo nombredirectorio.orig. Vaya al directorio de los fuentes y siga las instrucciones para construirlo. Use éstos para compilar. Si necesita editar algo, necesitará un parche. Una vez que lo tenga compilado, limpie el directorio fuentes.

Asegúrese que borra todos los archivos creados por algún guión de configuración. Haga entonces un cd hacia arriba, al directorio paterno. Deberá hacer algo como:

diff -uNr nombredirectorio.orig nombredirectorio >../SOURCES/nombredirectorio-linux.patch

Lo que creará un parche que podrá usar en su archivo spec. Advierta que el "linux" que aparece en el nombre del parche es sólo un identificador. Usted probablemente querrá usar algo más descriptivo como "config" o "bugs" para describir el porqué del parche. También es buena idea examinar el archivo parche que ha creado antes de usarlo para asegurarse de que no se han incluido binarios por error.

Creación de la lista de archivos

Ahora que tiene los fuentes listos para construir y sabe cómo hacerlo, constrúyalo e instálelo. Examine la salida de la secuencia de instalación y construya su lista de archivos a partir de ahí para incluirla en el archivo spec. Normalmente nosotros

construruimos el archivo spec en paralelo a todos estos pasos. Usted puede crear uno inicial y rellenar las partes sencillas e ir rellenando el resto conforme vaya completando los diferentes pasos.

Construyendo el paquete con RPM

Una vez que tenga su archivo spec, está listo para intentar construir su paquete. La forma más útil de hacerlo es con un comando como el siguiente:

bash$ rpm -ba foobar-1.0.spec (y con esto se construye el paquete RPM en su totalidad generando el archivo de instalación .rpm y el de fuentes src.rpm.

Hay otras opciones útiles con el parámetro -b tales como:

- p obliga a ejecutar solamente la sección prep del archivo spec.
- l efectúa algunos chequeos en %files.
- c ejecuta la sección %prep y compila. Resulta útil cuando no está seguro de si sus fuentes compilan completamente. Puede parecer inútil porque usted tal vez quiera jugar solamente con los fuentes hasta que compilen y usar entonces RPM, pero una vez que se haya acostumbrado a usar RPM, encontrará ocasiones en las que podrá usarla.
- i ejecuta las secciones %prep, %compile e %install.
- b ejecuta las secciones %prep, %compile e %install y construye el paquete binario.
- a ejecuta las secciones %prep, %compile e %install y construye los paquetes binario y fuentes.

6.8 Probándolo

Una vez que tenga los paquetes rpm binario y fuentes, necesitará probarlos. La mejor forma y la más sencilla es usar para el test una máquina completamente diferente de la que usó para la construcción. Después de todo, ha hecho un montón de make install en su máquina por lo que debería haber quedado instalado de forma aceptable.

Puede ejecutar un rpm -e nombredepaquete al paquete a probar(primero borrar), pero puede ser decepcionante porque en la construcción del paquete usted hizo un make install. Si se dejó algo fuera de la lista de archivos, no será desinstalado. Reinstale entonces el paquete binario y su sistema estará completo de nuevo, aunque no el paquete rpm. Asegúrese y tenga en mente que sólo porque usted hace rpm -ba package; la mayoría de la gente instalará su paquete sólo con rpm -i package.

Asegúrese también de que no hace nada en las secciones build o install que necesite hacerse cuando los binarios se instalan por sí mismos(con un make install).

6.9 ¿Qué hacer con los nuevos paquetes RPM?

Una vez que ha hecho su propio RPM de algo (asumiendo que no ha sido empaquetado en RPM con anterioridad), puede contribuir con su trabajo a otras personas (asumiendo igualmente que el paquete en RPM es libremente distribuible). Para hacerlo, puede querer subirlo a ftp.redhat.com o www.rpmfind.net, o algún otro sitio donde usted quiera compartir su trabajo.

6.10 ¿Y ahora qué?

Por favor mire las secciones anteriores sobre pruebas y qué hacer con los nuevos RPM. Queremos todos los paquetes RPM disponibles que podamos conseguir, y queremos que estén correctamente empaquetados. Por favor, tómese el tiempo de probarlos adecuadamente y de subirlos para el beneficio de todos.

También, por favor asegúrese de que no está haciendo llegar solamente software de libre disposición. Software comercial y shareware no deberían ser subidos a menos que esté claramente permitido en alguna cláusula de su licencia. Esto incluye el software de Netscape, ssh, pgp, etc.

7 Construcción multi-arquitectura de paquetes RPM

Ahora puede usarse RPM para construir paquetes para Intel i386, Digital Alpha ejecutando Linux y Sparc. También se ha informado de su funcionamiento en estaciones de trabajo SGI, HP, SCO y otras. Hay varias características que hacen que la construcción de paquetes para todas las plataformas sea fácil. La primera de éstas es la directiva "optflags" del archivo rpmrc. Puede usarse para asignar las opciones usadas durante la construcción del software con valores específicos de cada arquitectura. Otras son las macros "arch" en el archivo spec. Pueden usarse para diferentes cosas, en función de la arquitectura para la que se está construyendo. Otra más, es la directiva "Exclude" de la cabecera.

7.1 Macros

La macro %ifarch es muy importante para todo esto. La mayoría de las veces necesitará hacer un parche o dos específicos para una sóla arquitectura. En ese caso, RPM le permitirá aplicar ese parche sólo para una arquitectura.

En el ejemplo anterior, fileutils tiene un parche para máquinas de 64 bits. Obviamente, sólo tiene aplicación en Alpha, por el momento. Entonces, añadimos una macro %ifarch al parche de 64 tal como:

```
%ifarch axp
%patch1 -p1
%endif
```

Esto asegurará que el parche no es aplicado en cualquier arquitectura excepto en Alpha.

7.2 Excluyendo arquitectura de los paquetes

A la vez que puede tener fuentes RPM en un sólo directorio para todas las plataformas, hemos implementado la posibilidad de "excluir" paquetes para que no sean construidos en ciertas arquitecturas. Debido a esto, puede hacer cosas como:

rpm –rebuild /usr/src/SRPMS/*.rpm

y conseguir construir los paquetes adecuados. Si todavía no ha portado una aplicación a una determinada plataforma, todo lo que tiene que hacer es añadir una línea como:

ExcludeArch: axp

a la cabecera del archivo spec del paquete de fuentes. Reconstruya entonces el paquete sobre la plataforma para la que está preparado. Como resultado tendrá disponible un paquete compilable sobre Intel pero que es fácilmente omitible sobre Alpha.

8. Usando paquetes Debian - Convertiendo archivos .deb archivos .rpm

Debian es otra popular distribución de Linux. Existe para muchas plataformas como Intel, Alpha, Sparc, PowerPC, MIPS, Itanium. Debian usa el sistema de gestión de paquetes llamado DPKG. El sufijo de los paquetes debian es .deb. Lo especial del sistema de paquetes de debian es que se construyen estableciendo unas relaciones de dependencia realmente cuidadas de manera que el sistema es capaz de gestionar y resolver dichas dependencias de manera automática dando lugar a un sistema de instalación y mantenimiento de software realmente potente y cómodo.

Debian tiene cientos de paquetes con software muy útil que en una distribución basada en RPM como Red Hat , Suse o Mandrake desearíamos tener. En Debian hay la posibilidad de convertir los paquetes RPM a DEB con una herramienta llamada 'alien'. Pues bien esa misma herramienta la hay para Red Hat, Suse o Mandrake, buscala en http://www.rpmfind.net.

Con esa misma herramienta podemos convertir los DEB a RPM.

bash# rpm -i alien*.src.rpm (instalamos alien)

bash# alien -r mypaquete-debian.deb (esto debe crear el paquete rpm)

bash# rpm -i mypaquete.rpm (instalamos el paquete generado con alien)

9. Comparativa entre RPM y DEB

Tabla comparativa

Características	DEB	RPM
firma paquetes	sí	sí
checksums	sí	sí
permisos, dueños, etc	sí	sí
reconocible por archivo	sí	sí
desempaquetable por herramientas estandar	sí	no
metadata accesible por herramientas estandar	sí	no
creable con herramientas estandar	sí	no
dependencias	sí	sí
recomendaciones	sí	no
sugerencías	sí	no
conflictos	sí	sí
dependencias de versión y conflictos	sí	sí
dependencias de archivos	no	sí
copyright info	no	sí
grupo de paquetes	sí	sí
prioridad	sí	no
programas binarios	sí	no
pre-install	sí	sí
post-install	sí	sí
pre-remove	sí	sí
post-remove	sí	sí
verifica programas	no	sí
triggers	no	yes

Referencias

[1] http://kraus.gsi.dit.upm.es/doc/HOWTO/es/HOWTO/RPM-Como.html

[2] http://www.linuxdoc.org/HOWTO/RPM-HOWTO

[3] http://www.tldp.org/HOWTO/RPM-for-Unix-HOWTO.html

[4] http://kitenet.net/~joey/pkg-comp/

[5] http://www.cofradia.org/~aavelar/html/consol2003