RPM COMO

Donnie Barnes, djb@redhat.com
Traductor: Antonio Ismael Olea González,
olea@iname.com 2:345/108.9@fidonet.org V2.0, April 8, 1997

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

Índice General

1 Introducción

RPM es el gestor de paquetes de Red Hat (**R**ed Hat **P**ackage **M**anager). 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 binaria de forma que los ficheros 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 ficheros que puede ser usada para verificar paquetes e interrogarla para obtener información acerca de ficheros 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 aunque agradeceríamos informes de fallos (*bugs*) y sus reparaciones (*fixes*). Se concede permiso para usar y distribuir RPM, libre de *royalties*, bajo la protección de la licencia GPL.

Puede encontrar información más completa sobre RPM en el libro de Ed Bailey *Maximum RPM*. Dicho libro está disponible en `www.redhat.com`.

2 Visión general

Primero, permítame expresar parte de la filosofía tras RPM. Uno de los objetivos del diseño fue permitir el uso de fuentes "prístinas[1]".

Con RPP (nuestro anterior sistema de empaquetado del cual RPM *no deriva* en absoluto), nuestros paquetes de fuentes debían ser "hackeados[2]" para poder construir las aplicaciones desde ellos. Teóricamente, se podía instalar un paquete fuente RPP y efectuarle un `make` sin problemas. Pero los fuentes no eran las originales, y no había referencia alguna a los cambios que habíamos hecho para que pudieran compilar. Se hacía pues necesario bajarse los fuentes originales de forma separada.

Con RPM, tiene los fuentes originales junto al "parche[3]" que hemos usado para poder compilarlo. Vemos en esto una gran ventaja. ¿Por qué? Son varias las razones. La primera es que si sale disponible una nueva versión de un programa, usted no necesita empezar desde la nada para conseguir que compile bajo RHL. 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 ficheros. También puede encontrar fácilmente a qué paquete pertenece un fichero y de dónde proviene. Los ficheros RPM en sí

[1] N.T.: originales
[2] N.T.: retocados
[3] N.T.: *patch* en el original

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 fichero 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 fichero de configuración que usted tenga quedará a salvo.

Queremos agradecer a los colegas de la distribución BOGUS por muchas de sus ideas y conceptos que han sido incluidos en RPM. Aunque RPM está completamente escrito por Red Hat Software, su funcionamiento está basado en código escrito por BOGUS (PM y PMS).

3 Información general

3.1 Adquirir RPM

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

`ftp.redhat.com/pub/redhat/code/rpm`

3.2 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, AmigaOS, y otros. Queda advertido que los paquetes binarios generados en diferentes tipos de sistemas Unix no serán compatibles.

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.

4 Usando RPM

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

```
rpm -i foobar-1.0-1.i386.rpm
```

El siguiente comando más simple es desinstalar un paquete:

```
rpm -e foobar
```

Uno de los más complejos pero *más* útiles comandos le 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 fichero con un URL válido, como esto:

```
rpm -i ftp://ftp.pht.com/pub/linux/redhat/rh-2.0-beta/RPMS/foobar-1.0-1.i386.rpm
```

Apercíbase de que ahora 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*:

```
RPM version 2.3.9
Copyright (C) 1997 - 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}
```

Podrá encontrar más detalles acerca de la función de estos parametros en la página del manual de RPM.

5 Y ahora, ¿qué puedo hacer *de verdad* con RPM?

RPM es una herramienta potentísima y, como puede ver, dispone de varios parámetros. La mejor forma de apercibirse de ellas es examinando unos cuantos ejemplos. Antes mostramos una instalación/desinstalación sencilla, ahora van unos cuantos más:

- Supongamos que ha borrado unos cuantos ficheros 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:

```
rpm -Va
```

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

  ```
  rpm -qf /usr/X11R6/bin/xjewel
  ```

  La salida podría ser:

  ```
  xjewel-1.6-1
  ```

- Supongamos que acaba de hacerse con un nuevo paquete RPM de `koules`, pero no sabe qué puede ser. Para obtener información al respecto:

  ```
  rpm -qpi koules-1.2-2.i386.rpm
  ```

  La salida podría ser:

  ```
  Name        : koules                      Distribution: Red Hat Linux Colgate
  Version     : 1.2                               Vendor: Red Hat Software
  Release     : 2                             Build Date: Mon Sep 02 11:59:12 1996
  Install date: (none)                        Build Host: porky.redhat.com
  Group       : Games                         Source RPM: koules-1.2-2.src.rpm
  Size        : 614939
  Summary     : SVGAlib action game with multiplayer, network, and sound support
  Description :
  This arcade-style game is novel in conception and excellent in execution.
  No shooting, no blood, no guts, no gore.  The play is simple, but you
  still must develop skill to play.  This version uses SVGAlib to
  run on a graphics console.
  ```

- Ahora quiere saber qué ficheros instala el paquete RPM. Puede hacer:

  ```
  rpm -qpl koules-1.2-2.i386.rpm
  ```

  La salida es:

  ```
  /usr/doc/koules
  /usr/doc/koules/ANNOUNCE
  /usr/doc/koules/BUGS
  /usr/doc/koules/COMPILE.OS2
  /usr/doc/koules/COPYING
  /usr/doc/koules/Card
  /usr/doc/koules/ChangeLog
  /usr/doc/koules/INSTALLATION
  /usr/doc/koules/Icon.xpm
  /usr/doc/koules/Icon2.xpm
  /usr/doc/koules/Koules.FAQ
  /usr/doc/koules/Koules.xpm
  /usr/doc/koules/README
  /usr/doc/koules/TODO
  /usr/games/koules
  /usr/games/koules.svga
  /usr/games/koules.tcl
  /usr/man/man6/koules.svga.6
  ```

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.
- Haga un parche con todos los cambios que ha tenido que realizar para que los fuentes se compilen adecuadamente.
- Cree un fichero `.spec` para el paquete.
- Asegúrese de que todo está en su sitio.
- Construya el paquete usando RPM.

En general, RPM construirá tanto el paquete binario como el de los fuentes.

6.1 El fichero `rpmrc`

En lo sucesivo, la única configuración de RPM está disponible en el fichero `/etc/rpmrc`. Éste puede tener la siguiente apariencia:

```
require_vendor: 1
distribution: I roll my own!
require_distribution: 1
topdir: /usr/src/me
vendor: Mickiesoft
packager:  Mickeysoft Packaging Account <packages@mickiesoft.com>

optflags: i386 -O2 -m486 -fno-strength-reduce
optflags: alpha -O2
optflags: sparc -O2

signature: pgp
pgp_name: Mickeysoft Packaging Account
pgp_path: /home/packages/.pgp

tmppath: /usr/tmp
```

La línea `require_vendor` obliga a RPM a requerir una línea de vendor (distribuidor). Ésta puede venir o bien del propio fichero `/etc/rpmrc` o de la cabecera del fichero `.spec`. Para desactivar este parámetro debe cambiarse el número a `0`. Esto también se aplica a las líneas `require_distribution` y `require_group`.

La siguiente línea es la de `distribution`[4]. Puede definirla bien aquí o bien en el fichero `.spec`. Cuando se empaqueta para una distribución en concreto, es buena idea asegurarse de que esta línea es correcta, incluso cuando no se requiera. La línea de `vendor`[5] funciona de la misma manera, aunque puede contener cualquier cosa (ej.: *Joe's Software and Rock Music Emporium*).

[4]N.T.: distribución

[5]N.T.: distribuidor

RPM también soporta ahora el empaquetado de paquetes para múltiples arquitecturas. El fichero `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:

```
rpm --showrc
```

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

6.2 El fichero `spec`

Ahora hablaremos del fichero `.spec`. Estos ficheros son imprescindibles para construir un paquete. El fichero spec es una descripción del software acompañado con instrucciones sobre cómo construirlo y una lista de ficheros de todos los binarios instalados.

Debería nombrar a sus ficheros 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 fichero spec (`eject-1.4.spec`):

```
Summary: ejects ejectable media and controls auto ejection
Name: eject
Version: 1.4
Release: 3
Copyright: GPL
Group: Utilities/System
Source: sunsite.unc.edu:/pub/Linux/utils/disk-management/eject-1.4.tar.gz
Patch: eject-1.4-make.patch
Patch1: eject-1.4-jaz.patch
%description
This program allows the user to eject media that is autoejecting like
CD-ROMs, Jaz and Zip drives, and floppy drives on SPARC machines.

%prep
%setup
%patch -p1
%patch1 -p1

%build
make RPM_OPT_FLAGS="$RPM_OPT_FLAGS"

%install
install -s -m 755 -o 0 -g 0 eject /usr/bin/eject
install -m 644 -o 0 -g 0 eject.1 /usr/man/man1

%files
%doc README COPYING ChangeLog

/usr/bin/eject
/usr/man/man1/eject.1
```

6.3 La Cabecera

La cabecera 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 fichero rpm.
- `Version:` La cadena que vaya a ser el número de versión para el fichero 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 fichero `.gif` y estar situado en el directorio `SOURCES`.
- `Source:` Esta línea apunta a la localización HOME del fichero de fuentes original. Se usa si alguna vez quiere tener los fuentes de nuevo o chequear para nuevas versiones. Precaución: el nombre del fichero en esta línea DEBE coincidir con el nombre que tiene tal fichero en su sistema (ej.: no se haga con el fichero fuente y le cambie el nombre). Puede especificar más de un fichero fuente de esta forma:

  ```
  Source0: blah-0.tar.gz
  Source1: blah-1.tar.gz
  Source2: fooblah.tar.gz
  ```

  Estos ficheros deben residir en el directorio `SOURCES`. (La estructura de directorios es discutida en una sección posterior, "El Árbol de Directorios de las Fuentes", 6.9.1).
- `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 fichero de parches de esta forma:

  ```
  Patch0: blah-0.patch
  Patch1: blah-1.patch
  Patch2: fooblah.patch
  ```

  Estos ficheros 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, *public domain*[6], *distributable*[7] o *commercial*[8].
- `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`. Actualmente, esta jerarquía viene a ser:

  ```
  Applications            (aplicaciones)
      Communications      (comunicaciones)
      Editors             (editores)
          Emacs           (Emacs)
      Engineering         (ingenieria)
      Spreadsheets        (hojas de calculo)
      Databases           (bases de datos)
      Graphics            (graficos)
      Networking          (redes de comunicaciones)
      Mail                (correo smtp)
      Math                (matematicas)
      News                (noticias nntp)
  ```

[6] N.T.: dominio público
[7] N.T.: distribuible
[8] N.T.: comercial

```
        Publishing        (edicion)
            TeX           (TeX)
    Base                  (basico)
        Kernel            (nucleo)
    Utilities             (utilidades)
        Archiving         (archivo)
        Console           (consola)
        File              (ficheros)
        System            (sistema)
        Terminal          (terminales)
        Text              (texto)
    Daemons               (demonios)
    Documentation         (documentacion)
    X11                   (X11)
        XFree86           (XFree86)
            Servers       (servidores)
        Applications      (aplicaciones)
            Graphics      (graficos)
            Networking    (redes de comunicaciones)
        Games             (juegos)
            Strategy      (estrategia)
            Video         (video juegos)
        Amusements        (entretenimientos)
        Utilities         (utilidades)
        Libraries         (librerias)
        Window Managers   (gestores de ventana)
    Libraries             (librerias)
    Networking            (redes de comunicaciones)
        Admin             (administracion)
        Daemons           (demonios)
        News              (noticias nntp)
        Utilities         (utilidades)
    Development           (desarrollo)
        Debuggers         (depuradores)
        Libraries         (librerias)
            Libc          (libreria C)
        Languages         (lenguajes)
            Fortran       (fortran)
            Tcl           (tcl)
        Building          (Compilacion)
        Version Control   (control de versiones)
        Tools             (utiles)
    Shells                (interpretes de comandos)
    Games                 (juegos)
```

- `%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.

6.4 `%prep`

Esta es la segunda sección del fichero 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.

Una cosa a señalar: Cada una de estas secciones es sólo un lugar para ejecutar guiones de intérprete de comandos[9]. 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}${VERSIÓN}`, o cualquiera que use el fichero tar principal. (Apercíbese de que estas variables "$" *no* son variables reales dentro del fichero 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 fichero 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 fichero fuente `#` *después* de cambiar al directorio.
- `-T` Este parámetro anula la acción por defecto de desempaquetar el fichero fuente y requiere `-b 0` o `-a 0` para desempaquetar el fichero 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 fichero 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.

6.5 %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.

[9]N.T.: *shell scripts* en el original.

6.6 `%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 fichero `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.

6.7 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. 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`.

6.8 `%files`

Esta es la sección donde *debe* listar los ficheros del paquete binario. RPM no tiene forma de saber qué ficheros 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 ficheros 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 ficheros 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 ficheros en una sóla línea o puede listarlos de forma separada con una macro para cada uno de ellos.
- `%config` se usa para señalar los ficheros de configuración en un paquete. Ficheros así pueden ser `sendmail.cf`, `passwd`, etc. Si posteriormente desinstala un paquete que incluye ficheros de configuración, todos los ficheros sin modificar serán borrados y todos los ficheros modificados serán movidos a su nombre antiguo con el sufijo `.rpmsave` añadido a su nombre. También puede incluir múltiples ficheros 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 ficheros y posteriormente instalado como parte del paquete.
- `%files -f` <`nombredefichero`> le permitirá tener la lista de ficheros contenida en un fichero situado en el directorio de las fuentes. Resulta útil en los casos en los que un paquete puede crear su propia lista de ficheros por sí mismo. En ese caso sólo tendrá que incluir el nombre de ese fichero aquí y no necesitará especificar nada más.

La mayor precacución que debe tener en cuenta en la lista de ficheros es la inclusión de directorios. Si por accidente incluye `/usr/bin`, su paquete binario contendrá *todos* los ficheros contenidos en el directorio `/usr/bin` en su sistema.

6.9 Construcción

6.9.1 "El Árbol de Directorios de los Fuentes"

Lo primero que necesita es un árbol de directorios de compilación configurado de forma apropiada. Esto se puede hacer mediante el fichero `/etc/rpmrc`. La mayoría de la gente sólo usará `/usr/src`.

Puede que necesite crear los siguientes directorios para organizar un árbol de construcción:

- `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 ficheros fuente originales y los correspondientes parches. Es donde RPM buscará por defecto.
- `SPECS` es el directorio donde deben ir situados los ficheros spec.
- `RPMS` es donde RPM dejará los paquetes RPM binarios una vez construidos.
- `SRPMS` es donde RPM dejará los paquetes RPM fuentes.

6.9.2 Prueba de construcción

Lo primero que querrá hacer es asegurarse que los fuentes se construyen adecuadamente sin usar RPM. Para ello, desempaquete los fuentes, sitúese en el directorio `$NAME.orig`. De nuevo desempaquete los fuentes. Use éstos para compilar. Vaya al directorio de los fuentes y siga las instrucciones para construirlo. Si necesita editar algo, necesitará un parche. Una vez que lo tenga compilado, limpie el directorio fuentes.

Asegúrese que borra todos los ficheros 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 fichero 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 fichero parche que ha creado antes de usarlo para asegurarse de que no se han incluido binarios por error.

6.9.3 Creación de la Lista de Ficheros

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 ficheros a partir de ahí para incluirla en el fichero spec. Normalmente nosotros construimos el fichero 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.

6.9.4 Construyendo el paquete con RPM

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

```
rpm -ba foobar-1.0.spec
```

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

- `p` obliga a ejecutar solamente la sección `prep` del fichero 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.

Hay varios modificadores para el parámetro `-b`. Son los siguientes:

- `--short-circuit` El proceso de construcción comenzará directamente por la fase especificada, saltándose las previas a la indicada. (Sólo puede ser empleado en conjunción con `c` e `i`).
- `--clean` elimina el árbol de construcción una vez que ha concluido.
- `--keep-temps` mantendrá todos los ficheros temporales y los guiones que estuvieran en `/tmp`. Podrá saber qué ficheros fueron creados allí usando el parámetro `-v`.
- `--test` No lleva a cabo ninguna fase realmente, pero mantiene todos los ficheros temporales.

6.10 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 -u nombredepaquete` al paquete a probar, pero puede ser decepcionante porque en la construcción del paquete usted hizo un `make install`. Si se dejó algo fuera de la lista de ficheros, 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.

6.11 ¿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`.

6.12 ¿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[10] *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 y HP. 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 fichero `/etc/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 fichero 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 Ejemplo de fichero spec

El siguiente es parte del fichero spec para el paquete “`fileutils`”. Está configurado para compilar en Alpha e Intel.

```
Summary: GNU File Utilities
Name: fileutils
Version: 3.16
Release: 1
Copyright: GPL
Group: Utilities/File
Source0: prep.ai.mit.edu:/pub/gnu/fileutils-3.16.tar.gz
Source1: DIR_COLORS
Patch: fileutils-3.16-mktime.patch

%description
These are the GNU file management utilities.  It includes programs
to copy, move, list, etc, files.

The ls program in this package now incorporates color ls!

%prep
%setup

%ifarch alpha
%patch -p1
autoconf
%endif
%build
configure --prefix=/usr --exec-prefix=/
make CFLAGS="$RPM_OPT_FLAGS" LDFLAGS=-s
```

[10] N.T.: Probar antes de comprar

```
%install
rm -f /usr/info/fileutils*
make install
gzip -9nf /usr/info/fileutils*

.
.
.
```

7.2 Optflags

En este ejemplo puede ver cómo se usa la directiva "`optflags`" desde el fichero `/etc/rpmrc`. En función de la arquitectura para la que está construyendo, el valor adecuado lo proporciona `RPM_OPT_FLAGS`. Debe parchear el fichero `Makefile` de su paquete para usar esta variable en lugar de las directivas normales que puede usar (como `-m486` y `-O2`). Tendrá una mejor perspectiva de lo que necesita hacer instalando este paquete de fuentes, desempaquetando el fichero tar con los fuentes y examinando el fichero `Makefile`. Examine el parche para el `Makefile` y compruebe qué cambios son necesarios realizar.

7.3 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.

7.4 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 fichero 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.

7.5 Acabando

Usar RPM para crear paquetes para múltiples arquitecturas es generalmente más sencillo de hacer que conseguir que el paquete compile por sí mismo en todos los casos. Como siempre, la mejor ayuda disponible cuando uno se queda bloqueado al construir un paquete RPM es examinar un paquete de fuentes similar.

8 Copyright

9 Anexo: El INSFLUG

El *INSFLUG* forma parte del grupo internacional *Linux Documentation Project*, encargándose de las traducciones al castellano de los Howtos (Comos), así como la producción de documentos originales en aquellos casos en los que no existe análogo en inglés.

En el **INSFLUG** se orienta preferentemente a la traducción de documentos breves, como los *COMOs* y *PUFs* (**P**reguntas de **U**so **F**recuente, las *FAQs*. `:)`), etc.

Diríjase a la sede del INSFLUG para más información al respecto.

En la sede del INSFLUG encontrará siempre las **últimas** versiones de las traducciones: `www.insflug.org`. Asegúrese de comprobar cuál es la última versión disponible en el Insflug antes de bajar un documento de un servidor réplica.

Se proporciona también una lista de los servidores réplica (*mirror*) del Insflug más cercanos a Vd., e información relativa a otros recursos en castellano.

Francisco José Montilla, `pacopepe@insflug.org`.