Preparación para el examen LPI 101

Tema 102.6 Administrando paquetes RPM

Créditos y licencia de uso

Coordinación:
Manuel Guillán (xLekOx) lpi@xlekox.org

Traducción:
Juan Maria Gil (Smooth) yo@juanmaria.com
Ivan Servia (katas) ivanservia@hotmail.com

Maquetación:
Manuel Guillán (xLekOx) lpi@xlekox.org
Javier Pulido (jpulido) javier.pulido@wanadoo.es

Versión 1.0 (26-09-2004 14:00)

ÍNDICE

Índice de contenido

Introducción

En este capítulo se verá como administrar un sistema usando el administrador de paquetes RPM. Esto incluye comandos para instalar, actualizar y desinstalar programas, así como otras características, como saber la versión instalada, contenidos, dependencias, integridad del paquete, etc.

Los comandos que se verán en este tema son:

- rpm
- grep

Se verá la configuración del fichero:

- /etc/rpmrc
- /usr/lib/rpm/*

Este tema tiene un peso (importancia) de 8 de cara al examen final de la certificación LPI 101. El total de la suma de pesos de todos los temas es de 106.

Gestor de paquetes Red Hat

Hay que familiarizarse con las siguientes tareas: instalación del paquete, desinstalación del paquete, determinar la versión de dicho paquete y saber la versión del software que contiene, listar los archivos del paquete, listar los archivos de documentación de paquete, listar los archivos de configuración o los scripts de instalación o desinstalación, encontrar un determinado paquete que está instalado, saber que paquetes han sido instalados en el sistema (todos o un grupo determinado), averiguar en que paquete puede encontrarse un programa o archivo, verificar la integridad del paquete, saber la firma PGP o GPG del paquete y actualizarlo.
Requiere el uso de los comandos y programas: rpm, grep.

En ocasiones es difícil obtener el pedazo de código necesario para compilar correctamente el sistema, y a menos que se desee realizar cambios o leer el código, no existe una ventaja real en hacerlo. La mayoría del software está distribuido en formato binario, es decir, ya compilado.

El más popular gestor de paquetes que se utiliza con Linux es RPM, o Red Hat Package Manager. A pesar de ser creado por Red Hat, se utiliza en la mayoría de las distribuciones por defecto exceptuando Slackware, Debian, Gentoo... RPM es una de las contribuciones de Red Hat a la comunidad de software libre más conocidas y una de las que ahorraron a muchos administradores tiempo y esfuerzo.

Un sistema de gestión de paquetes mejora la distribución binaria gestionando el control de la versión, las dependencias con otros paquetes y su administración. Utilizando las herramientas del paquete, se puede comprobar la versión instalada, los archivos incluidos en el paquete, etc. RPM está compuesto por:

- Archivos del paquete (`*.rpm`)
- La base de datos RPM
- La herramienta `rpm`

Archivos del paquete (*.RPM)

Los archivos RPM se distribuyen para la mayoría de las aplicaciones. Un archivo RPM incluye las siguientes partes:

- Archivos de la aplicación comprimidos
- Nombre y versión del paquete
- Fecha de realización y fecha de publicación
- Descripción del paquete y de la aplicación
- Información de quién realizó el paquete
- MD5 “checksum” para verificar la integridad del paquete
- Otros paquetes requeridos (dependencias)

Como se puede observar, dentro de un paquete RPM se incluye mucha información. A través de los distintos archivos, se incluye toda la información necesaria para instalar y mantener el paquete. Los RPM siguen la siguiente tipología estándar:

package-version-patch.architecture.rpm

donde:

- package - Nombre de la aplicación instalada por el paquete.
- versión - Número de la versión de la aplicación.
- match - Número de "arreglo" del paquete. Si se produce un pequeño cambio o el administrador realiza una modificación en el paquete, este número se incrementa.
- architecture - la arquitectura del computador para la cual está realizado el paquete. Esto es muy importante ahora que Linux se ejecuta en tantas computadoras distintas. Algunos ejemplos: i386, i586, y i686 para Intel x86 y compatibles; sparc para Sun Sparc,; y alpha para Digital/Compaq Alpha.

Ejemplo:

ethereal -0.8.9-1.i386.rpm

Este paquete contiene la versión 0.8.9 de Ethereal, un paquete "sniffer" utilizado para reiniciar una red. Esta es la primera construcción de este paquete, y es para la plataforma i386 (Intel PC).

Un lugar para encontrar paquetes rpm de muchas aplicaciones es `www.rpmfind.net`

La base de datos RPM

La información sobre todos los paquetes instalados en el sistema se mantiene en una base de datos. Ésta se encuentra en el directorio `/var/lib/rpm`. Estos datos se utilizan para encontrar las dependencias, comprobar los ficheros que ya existen y verificar los paquetes instalados. Siempre que se utiliza el comando rpm se consulta la base de datos.

Ocasionalmente la base de datos tendrán inconsistencias y será necesario reconstruirla utilizando el comando `rpm` con `–rebuilddb`:

```
#rpm –rebuilddb
```

Cuando sucede una inconsistencia o se corrompe la base de datos, se reciben errores extraños al añadir o borrar paquetes.

La herramienta rpm

La herramienta rpm es una herramienta de la línea de comandos que se preocupa de RPM, por lo tanto no cabe confusión.

El comando `rpm` se utiliza para:

- Instalar paquetes
- Actualizar paquetes
- Borrar y desinstalar paquetes
- Consultar la base de datos RPM en busca de información
- Verificar el archivo del paquete
- Comprobar los archivos instalados
- Construir un paquete binario desde el código

La herramienta rpm actualiza errores del sistema realizando la instalación, desinstalación o actualización. Esto ayuda a proteger contra la instalación de un paquete inútil o dañino para otra aplicación. Estos diagnósticos incluyen la comprobación de:

- Suficiente espacio libre para el paquete
- Los archivos existentes no serán sobrescritos
- Todas las dependencias están instaladas.

Validando la integridad del paquete

RPM incluye funciones que permiten comprobar la integridad de un paquete para confirmar que ha sido correctamente descargado y no es falso. Eso se realiza utilizando el algoritmo MD5 y la herramienta GnuPG. MD5 es un algoritmo matemático que se utiliza para comprobar que un archivo no se ha modificado. Cuando se comprueba el archivo, el algoritmo extrae un número de comprobación (checksum), y si este número coincide con el generado por el archivo antes de la descarga, el archivo no está corrupto. La herramienta GnuPg es un paquete de encriptación de llave pública que puede ser usado para comprobar la autenticidad del código de un archivo o documento y encriptar las comunicaciones. GnuPG se instala por defecto en los sistemas Red Hat.

Las opciones `-k` o `--checksig` comprueban la integridad de un paquete utilizando MD5 y/o GnuPG.

```
rpm -k package_file.rpm
```

Para que funcione correctamente, se deben seguir los siguientes pasos:

1.- Instalación the la aplicación GnuPG si no está instalada. Se puede conseguir desde `www.gnupg.org`.

2.- Recuperación de la llave pública desde el administrador de la aplicación que se desee comprobar. Esto está generalmente disponible en los sitios Web o FTP. Por ejemplo, Red Hat está disponible en su FTP SITE y se denomina RPM-GPG-KEY.

3.- Añadir la llave pública apropiada al conjunto de llaves utilizando el comando gpg -import. Por ejemplo:

```
# gpg --import RPM-GPG-KEY
```

(se omite la salida por pantalla intencionadamente)

Si el paquete se valida correctamente, rpm escribirá un mensaje similar al siguiente:

```
# rpm -K wget-1.5.3-6.src.rpm
wget-1.5.3-6.src.rpm: md5 gpg OK
```

Si el paquete no se valida, el mensaje será el siguiente:

```
# rpm -K wget-1.5.3-10.i386.rpm
wget-1.5.3-10.i386.rpm: rpmReadSignature failed
```

Algunos paquetes utilizan PGP para comprobar la integridad mientras otros utilizan GnuPG. PGP se puede descargar desde `www.pgpi.com` .

Instalando Paquetes

El Gestor de Paquetes Red Hat (Red Hat Package Manager o RPM) simplifica muchísimo las instalaciones de software nuevo. Añadir un nuevo paquete al sistema puede ser tan simple como escribir el siguiente comando:

rpm -i *fichero_paquete*.rpm
o:
rpm --install *fichero_paquete*.rpm

CONSEJO PARA EL EXAMEN: Muchas de las opciones disponen de una versión corta y otra larga, por ejemplo, -i y --install. Debemos asegurarnos de conocer las dos versiones en el examen y recordar si van en mayúsculas o minúsculas.

La herramienta rpm también es capaz de instalar varios paquetes al mismo tiempo. Por ejemplo:

rpm -i *primer_fichero_paquete*.rpm *segundo_fichero_paquete*.rpm *tercer_fichero_paquete*.rpm
o:
rpm -i *.rpm

Se pueden utilizar comodines para instalar paquetes, pero no para eliminarlos.

La instalación simultánea de varios paquetes puede ser práctica cuando cada paquete va siendo instalado después de aquellos de los que depende. Para cada paquete instalado se realizan las verificaciones estándar de coherencia. Hay que tener en cuenta que el fallo en la instalación de un solo paquete podría abortar la ejecución de todo el comando y hacer que no se instalase ninguno.

La herramienta rpm también puede obtener paquetes desde Internet, lo que nos ahorraría uno o dos pasos en la instalación. El formato de la dirección del paquete es el mismo que utilizaríamos en un navegador web. La herramienta soporta tanto FTP anónimo como autorizado. Por ejemplo:

```
rpm -i ftp://rpmfind.net/linux/redhat/redhat-7.0/i386/en/RedHat/RPMS//libpcap-0.4-29.i386.rpm
```
o:
```
rpm -i http://rpmfind.net/linux/redhat/redhat-7.0/i386/en/RedHat/RPMS//libpcap-0.4-29.i386.rpm
```

Al instalar paquetes podemos añadir algunas opciones prácticas como -v y -h (--hash).
La opción -v muestra el nombre del paquete que está siendo instalado.

```
# rpm -iv libpcap-0.4-19.i386.rpm
libpcap-0.4-19
```

La opción -h va mostrando caracteres '#' (hash) mientras el paquete se está instalado para indicar la marcha de la instalación. Esto puede ser práctico cuando instalemos paquetes muy grandes.

```
# rpm -ivh libpcap-0.4-19.i386.rpm
```

```
libpcap
##################################################
```

Como se comentó anteriormente, rpm verifica todas las dependencias necesarias antes de instalar o eliminar un paquete. Si se encontrase algún problema, se mostraría un mensaje de error parecido al siguiente:

```
# rpm -ivh ethereal-0.8.9-1.i386.rpm
error: failed dependencies:
libpcap >= 0.4 is needed by ethereal-0.8.9-1
```

Esto significa que el paquete *ethereal-0.8.9-1.i386.rpm* necesita que esté instalado el paquete *libpcap* de versión 0.4 o mayor.

Si un paquete intentase sobreescribir un fichero existente, se mostraría un mensaje de error parecido a este:

```
# rpm -ivh libpcap.rpm
file /usr/lib/libpcap.a from install of libpcap-0.4a6-35 conflicts with file from package
libpcap-0.4-19
file /usr/man/man3/pcap.3.gz from install of libpcap-0.4a6-35 conflicts with file from
package libpcap-0.4-19
```

Por defecto, rpm no sobreescribirá ficheros de otros paquetes. En caso de que existiese una buena razón para continuar con la operación a pesar de los mensajes de advertencia, podrían utilizarse algunas de las opciones de sobreescritura que proporciona rpm y que se muestran a continuación:

--force—Fuerza a rpm a sobreescribir ficheros o paquetes existentes.
--nodeps—Se salta la verificación de dependencias. Esto es práctico si el paquete del que depende la instalación ha sido ya instalado mediante otro método como, por ejemplo, compilándolo a partir de los fuentes.
--replacefiles—Sobreescribe ficheros propiedad de otros paquetes.

Siguiendo el ejemplo anterior, asumamos que la librería *libpcap* ha sido compilada directamente desde los fuentes y no instalada a partir de un RPM. Para instalar el paquete Ethereal habría que utilizar el siguiente comando:

```
rpm -ivh --nodeps ethereal-0.8.9-1.i386.rpm
```

En esta ocasión, el paquete se instalará sin que aparezca ningún error. De todas formas, para que la aplicación funcione correctamente, la dependencia necesaria tiene que estar instalada correctamente. Saltarse los avisos es una práctica arriesgada que hay que tomar con mucha precaución considerando las posibles consecuencias, sobre todo en paquetes importantes.

Actualizando Paquetes

Si en el mundo de software hay algo seguro, es que continuamente nos encontraremos con

actualizaciones y nuevas versiones de los paquetes. RPM facilita esta labor realizando la eliminación de la versión antigua y la instalación de la nueva en un solo paso. Para actualizar un paquete simplemente utilizaremos la opción -U:

```
rpm -U fichero_paquete.rpm
```

o:

```
rpm --upgrade fichero_paquete.rpm
```

De la misma forma que con las opciones de instalación, en este caso también es recomendable el uso de los parámetros -v y -h.

Como la opción de actualización empieza eliminando la versión anterior, se guardarán los ficheros de configuración que contengan la extensión *.rpmsave*. Después, en la segunda fase se instalará la nueva versión realizándose en este paso las verificaciones estandar. Si no hubiese ninguna versión antigua del paquete en el sistema, rpm continuará adelante e instalará la nueva versión como si de una instalación nueva se tratase. Muchos administradores utilizan siempre la opción -U para cualquier tipo de instalación, de esta forma tienen los dos casos cubiertos ya que rpm intentará primero actualizar.

Otra opción práctica es -F (o --freshen), la cual actualiza un paquete sólo si ya existe una versión anterior instalada. De esta forma, se pueden actualizar un buen número de paquetes de una sola vez.

```
rpm -Fvh *.rpm
```

Este comando intentará actualizar cualquier paquete ya instalado en el sistema del que exista un fichero de paquete nuevo en el directorio actual. Combinando esto con el hecho de que la mayoría de las distribuciones, mantienen en sus sitios web un directorio con los paquetes que han sido actualizados desde la versión inicial, nos encontramos con una buena técnica para mantener nuestros sistemas actualizados.

Desinstalando Paquetes

Para eliminar un paquete del sistema se utilizan las opciones -e o –uninstall:

```
rpm -e nombre_paquete
```

o:

```
rpm --uninstall nombre_paquete
```

A la hora de desinstalar un paquete, debemos recordar que hemos de utilizar el nombre del paquete y no el nombre del fichero que contenía.
También hay que tener en cuenta que los comodines no funcionan en la eliminación de paquetes.
Las dos opciones realizan la misma función, simplemente, algunas personas encuentran que la opción -uninstall es más fácil de recordar.
Durante la desinstalación de un paquete, rpm realiza los chequeos de dependencias habituales y, por defecto, no permitirá desinstalar un paquete si existiese otro dependiente de éste.

```
# rpm -e libpcap
error: removing these packages would break dependencies:
libpcap >= 0.4 is needed by ethereal-0.8.9-1
```

Podemos utilizar la opción --nodeps para saltarnos estos avisos.
Cuando se desinstala un paquete, rpm guarda todos los ficheros de configuración modificados. De esta forma, si más adelante volviésemos a reinstalar el paquete no tendríamos que volverlo a configurar.

```
# ls /etc/pine*
/etc/pine.conf
# rpm -e pine
# ls /etc/pine*
/etc/pine.conf.rpmsave
```

Consultando la base de datos de RPM

La base de datos de RPM se guarda en /var/lib/rpm y contiene información sobre cada paquete instalado en el sistema. Estos datos pueden ser consultados para recopilar información práctica para la administración del sistema. Todas las opciones de consulta se realizan con el comando rpm y la opción -q.

Listando los paquetes instalados

La consulta más básica que podemos hacer es la de la versión de un paquete instalado:

rpm -q *nombre_paquete*

o:

rpm --query *nombre_paquete*

Por ejemplo:

```
# rpm -q kernel
kernel-2.2.14-5.0
```

Para listar todos los paquetes instalados en el sistema se utiliza la opción -a:

rpm -qa

Se puede combinar esta opción con *grep* para ver todos los paquetes instalados de un determinado grupo o familia.
El siguiente ejemplo utiliza rpm -qa para encontrar todos los paquetes y *grep* busca en esta lista todos los que contengan la palabra *kernel*:

```
# rpm -qa | grep kernel
kernel-headers-2.2.14-5.0
kernel-2.2.14-5.0
kernel-pcmcia-cs-2.2.14-5.0
kernel-utils-2.2.14-5.0
```

Averiguando que paquete instaló un determinado fichero

Hay ocasiones en las que no estamos seguros de que paquete instaló un determinado fichero y necesitamos averiguarlo. La opción -f nos proporciona ésta información.

rpm -qf *filename*

Por ejemplo:

```
# rpm -qf /etc/bashrc
bash-1.14.7-22
```

Esto nos indica que el fichero *bashrc* fue instalado por el paquete *bash-1.14.7-22*.

Listando los ficheros de un paquete

Para listar todos los ficheros instalados por un paquete se utiliza la opción -l.

rpm -ql *nombre_paquete*

Por ejemplo, para listar todos los ficheros del paquete openssh-clients teclearíamos lo siguiente:

```
# rpm -ql openssh-clients
/etc/ssh/ssh_config
/usr/bin/slogin
/usr/bin/ssh
/usr/bin/ssh-add
/usr/bin/ssh-agent
/usr/man/man1/slogin.1.gz
/usr/man/man1/ssh-add.1.gz
/usr/man/man1/ssh-agent.1.gz
/usr/man/man1/ssh.1.gz
```

Para listar los ficheros que serán instalados con el paquete se utiliza adicionalmente la opción -p.

rpm -qlp *fichero_paquete.rpm*

Mostrando información de un paquete

Para ver la descripción y otras informaciones sobre un determinado paquete se utiliza la opción -i.

rpm -qi *nombre_paquete*

Por ejemplo, para obtener información sobre el kernel Linux instalado en nuestro sistema, escribiríamos lo siguiente:

```
# rpm -qi kernel
```

(... a continuación se muestra una información muy completa sobre el paquete ...)

Si quisiéramos obtener ésta misma información pero de un paquete que aun no ha sido instalado, utilizaríamos la opción -p:

rpm -qip *fichero_paquete*.rpm

Mostrando los Scripts de un paquete

Algunos paquetes incluyen scripts (ficheros de comandos shell) que se ejecutan antes o después de la instalación de los mismos. Para ver estos scripts se utiliza el siguiente comando:

rpm -qp --scripts *fichero_paquete*.rpm

Por ejemplo, para ver los scripts que se ejecutarán cuando el paquete *kernel-2.2.14-5.0.i686.rpm* sea instalado, escribiríamos lo siguiente:

```
# rpm -qp –scripts /mnt/cdrom/RedHat/RPMS/kernel-2.2.14-5.0.i386.rpm
```

Verificando ficheros de paquetes

A veces es necesario verificar si un fichero ha cambiado desde la instalación de un paquete. Podríamos haber hecho algunos cambios en algún fichero de configuración y ahora necesitar averiguar cuales fueron los ficheros modificados. También podría interesarnos verificar si los ficheros extraídos de determinados paquetes han sido modificados por hackers o por algún virus. La herramienta rpm nos proporciona esta funcionalidad a través de la opción de verificación -V.

rpm -V *nombre_paquete*

Cuando realiza una verificación, rpm realiza una serie de comprobaciones en los ficheros instalados y muestra aquellos que han cambiado desde la instalación. Cada fichero tiene una entrada asociada en la base de datos RPM donde se almacena una firma MD5, el tamaño del mismo, un puntero a un enlace simbólico (symbolic link), la fecha y hora de modificación, el usuario y grupo propietarios y los permisos y tipo del fichero. Si el fichero fuese un dispositivo, también se comprueban los números de dispositivo mayor y menor. La siguiente tabla muestra la salida del comando según que característica haya sido cambiada.

Tabla 6-1 Características de verificación de paquetes

Ítem	*Significado*
.	No se ha encontrado ningún cambio en esta característica
5	Cambio en la firma MD5
S	Cambio en el tamaño del fichero
L	Cambio en el enlace simbólico (Symbolic link)

Ítem	*Significado*
T	Cambio en fecha u hora de modificación
G	Cambio en grupo propietario
U	Cambio en usuario propietario
D	Cambio en los números mayor/menor de dispositivo
M	Cambio en permisos y/o tipo de fichero

Por ejemplo, para ver que ficheros de configuración instalados por el paquete han sido modificados desde la instalación, escribiríamos lo siguiente:

```
# rpm -V setup
S.5....T c /etc/hosts.allow
......G. c /etc/profile
S.5....T c /etc/services
```

Esta salida nos muestra que han cambiado el tamaño, firma MD5 y hora de modificación en los ficheros *hosts.allow* y *services*. En cambio, al fichero *profile* sólo se le ha cambiado el grupo propietario.

Para verificar un paquete a partir del nombre de fichero del paquete, se utiliza la opción -p.

rpm -Vp *fichero_paquete*.rpm

Se pueden verificar todos los paquetes instalados en el sistema mediante la opción -a. Esta es una forma rápida de averiguar que ficheros han cambiado desde que se instaló el sistema.

rpm -Va

Creando paquetes binarios a partir de paquetes de fuentes

No todos los paquetes RPM distribuyen ficheros binarios. Algunos distribuyen el código fuente y los scripts de instalación que permiten la creación de paquetes RPM binarios personalizados, de esta forma podemos cambiar u optimizar un determinado paquete para nuestras necesidades o hardware particular.
Estos paquetes utilizan un sistema de nombres ligeramente diferente ya que no dependen de una arquitectura de sistema en particular.

paquete-version-patch.src.rpm

Los RPMs de fuentes se instalan, igual que los binarios, con el comando *rpm -i*.
La instalación coloca los diferentes componentes del paquete dentro de la jerarquía de directorios /usr/src/redhat.
La siguiente tabla muestra la función de los directorios de este árbol.

Tabla 6-2 Subdirectorios en el árbol /usr/src/redhat

Directorio	***Función***
/usr/src/redhat/SOURCES	Contiene el código fuente de la aplicación
/usr/src/redhat/SPECS	Contiene el fichero spec de la aplicación
/usr/src/redhat/BUILD	Donde se compila en código fuente
/usr/src/redhat/RPMS	Contiene el RPM binario final
/usr/src/redhat/SRPMS	Contiene el RPM fuente creado por el proceso

El fichero spec de un paquete controla como se compila el mismo y que scripts se ejecutarán cuando sea instalado o desinstalado.
El nombre de este fichero es /usr/src/redhat/SPECS/nombre_paqute.spec.
El fichero spec tiene ocho secciones como se muestra en la siguiente tabla:

Tabla 6-3 Secciones del fichero spec

Sección	***Descripción***
Header (Cabecera)	Información general, nombre, versión, etc
Prep (Preparación)	Scripts que realizarán las tareas necesarias antes del proceso de compilación
Buid (Construcción)	Comandos para construir el fichero spec y para compilar el código fuente
Install (Instalación)	Comandos necesarios para instalar el software en un sistema
Clean (Limpieza)	-Opcional- Comandos para limpiar el entorno de compilación en el caso de que se volviese a generar éste paquete
Optional Install Uninstall Scripts (Scripts opcionales de instalación y desinstalación)	Scripts opcionales que se pueden ejecutar en el proceso de instalación y desinstalación del paquete
File List (Lista de ficheros)	Lista de ficheros del paquete
Changelog (Registro de cambios)	Registro de los cambios que se hagan en el paquete

CONSEJO PARA EL EXAMEN: Debemos saber como construir un paquete binario a partir de un paquete de fuentes, incluyendo las modificaciones del fichero *spec*.

Esta es una sección **build** de ejemplo sacada del fichero spec de la aplicación *wget*:

```
%build
#./configure --prefix=/usr --sysconfdir=/etc
%configure --sysconfdir=/etc
make
```

Si quisiéramos hacer algunos cambios en el proceso de compilación deberíamos hacerlos aquí. Esto es lo que nos permite personalizar el paquete de forma que se adapte a nuestras necesidades. Una vez que hemos realizado todas las modificaciones oportunas hay que construir el paquete binario. Esto se hace con *rpm* y la opción -ba.

```
rpm -ba package.spec
```

Una vez completado el proceso, el paquete binario quedará en el directorio /usr/src/redhat/RPMS.

Ejemplos prácticos

Ejemplo 1
Para instalar un paquete nuevo, simplemente se utiliza el comando *rpm -i* seguido del nombre de un archivo de paquete. Si el nuevo paquete dependiese de otro paquete, la instalación daría un error como el siguiente:

```
# rpm -iv netscape-communicator-4.72-3.i386.rpm
error: failed dependencies:
netscape-common = 4.72 is needed by
netscape-communicator-4.72-3
```

Para corregir este tipo problemas, debemos instalar primero aquellos paquetes que satisfagan esas dependencias, en este ejemplo, *netscape-communicator* depende de *netscape-common*, por tanto debemos instalar primero este último paquete:

```
# rpm -iv netscape-common-4.72-3.i386.rpm
netscape-common

# rpm -iv netscape-communicator-4.72-3.i386.rpm
netscape-communicator
```

Ejemplo 2
Se puede actualizar un paquete existente a una nueva versión mediante la opción –U. El modo de actualización es realmente es un caso especial del modo de instalación, donde los paquetes existentes pueden ser reemplazados por las nuevas versiones. La opción -U instalará un paquete incluso si este no estuviese ya instalado, en este caso se comportaría de la misma forma que la opción -i:

```
# rpm -U netscape-common-4.72-3.i386.rpm
```

Ejemplo 3:
La eliminación de un paquete es lo contrario de la desinstalación del mismo y, por tanto, tiene las mismas restricciones de dependencia:

```
# rpm -e netscape-common
error: removing these packages would break dependencies (la eliminación de estos paquetes
        rompería algunas dependencias):
netscape-common = 4.72 is needed by
netscape-communicator-4.72-3
```

Ejemplo 4
Para determinar la versión del software contenido en un fichero RPM se utilizan las opciones de consulta e información del paquete:

```
# rpm -qpi xv-3.10a-13.i386.rpm | grep Version
Version : 3.10a Vendor: Red Hat Software
```

Cuando los paquetes estén ya instalados, se omite la opción -p y se especifica el nombre del paquete en lugar del nombre de fichero del paquete:

```
# rpm -qi kernel-source | grep Version
Version : 2.2.5 Vendor: Red Hat Software
```

Ejemplo 5

Utilizar el modo de consulta y listar los ficheros contenidos en un paquete:

```
# rpm -qlp gnucash-1.3.0-1.i386.rpm
/usr/bin/gnc-prices
/usr/bin/gnucash
/usr/bin/gnucash.gnome
/usr/doc/gnucash
/usr/doc/gnucash/CHANGES
(...el listado continúa ...)
```

Para un paquete ya instalado, entrar en modo consulta y utilizar la opción -l además del nombre del paquete:

```
# rpm -ql kernel-source
/usr/src/linux-2.2.5/COPYING
/usr/src/linux-2.2.5/CREDITS
/usr/src/linux-2.2.5/Documentation
/usr/src/linux-2.2.5/Documentation/00-INDEX
/usr/src/linux-2.2.5/Documentation/ARM-README
(...el listado continúa ...)
```

Ejemplo 6
Listar los ficheros de documentación en un paquete:

```
# rpm -qd at
/usr/doc/at-3.1.7/ChangeLog
/usr/doc/at-3.1.7/Copyright
/usr/doc/at-3.1.7/Problems
/usr/doc/at-3.1.7/README
/usr/doc/at-3.1.7/timespec
/usr/man/man1/at.1
/usr/man/man1/atq.1
/usr/man/man1/atrm.1
/usr/man/man1/batch.1
/usr/man/man8/atd.8
/usr/man/man8/atrun.8
```

Para indicar el nombre de fichero del paquete se usa la opción –p.

Ejemplo 7

Listar los scripts o ficheros de configuración de un paquete:

```
# rpm -qc at
/etc/at.deny
/etc/rc.d/init.d/atd
```

Ejemplo 8

Determinar desde que paquete en particular fue instalado un determinado fichero. Por supuesto, no todos los ficheros proceden de paquete:

```
# rpm -qf /etc/issue
file /etc/issue is not owned by any package (este mensaje indica que ese fichero no es
        propiedad de ningún paquete)
```

Para aquellos ficheros que son miembros de algún paquete, el resultado será parecido al siguiente:

```
# rpm -qf /etc/aliases
sendmail-8.9.3-10
```

Ejemplo 9

Listar los paquetes que han sido instalados en el sistema (todos o un subconjunto):

```
# rpm -qa
```
(... a continuación se listarán centenares de paquetes ...)

Ejercicios TEST

1. ¿Que sistema de paquetización utiliza Red Hat?
   A. rpm
   B. deb
   C. tgz
   D. rhp

2. ¿Que opción de *rpm* se utiliza cuando se reciben mensajes extraños que hacen suponer que la base de datos rpm está corrupta?
   A. rpm –fixdb
   B. rpm –rebuilddb
   C. rpm –updatedb
   D. rpm –regendb

3. ¿Que métodos de control de integridad de paquetes soporta RPM? (Selecciona todos los que procedan)
   A. MD5
   B. 3DES
   C. PGP
   D. GnuPG

4. Hay que utilizar el comando _____ para instalar el paquete llamado *processor-4.2.i386.rpm*.

5. ¿Que comando o comandos se utilizan para eliminar un paquete RPM?
   A. rpm --uninstall <nombre_de_paquete>
   B. rpm --remove <nombre_de_paquete >
   C. rpm -e <nombre_de_paquete >
   D. rpm -u <nombre_de_paquete >

6. ¿Cual de estos ficheros indica como fue compilado un paquete fuente RPM?
   A. Makefile
   B. fichero spec
   C. fichero config
   D. fichero .conf

Respuestas TEST

1. A. El sistema de paquetización RPM fue creado por Red Hat. Debian utiliza los paquetes .deb.

2. **B.** En ocasiones puede corromperse la base de datos RPM, en estos casos el comando *rpm --rebuilddb* intentará reconstruir la base de datos. El resto de las opciones son incorrectas.

3. **A, C, y D.** RPM soporta éstos tres métodos.

4. Tanto rpm -i processor-4.2-i386.rpm como rpm --install processor-4.2-i386.rpm son correctas.

5. **A y C.** Tanto *rpm --uninstall* como *rpm -e* eliminan paquetes RPM. El resto de las opciones son incorrectas.

6. **10. B.** El fichero *spec* es el que contiene las opciones de compilación. Se utiliza un Makefile para compilar código que no está en formato RPM.

Bibliografia y enlaces recomendados

LPIC 1 Certification Bible (Bible) by Angie Nash, Jason Nash
John Wiley & Sons; Bk&CD-Rom edition (July 1, 2001) ISBN: 0764547720

LPI Linux Certification in a Nutshell by Jeffrey Dean
O'Reilly & Associates; 1st ed edition (May 15, 2001) ISBN: 1565927486

CramSession's LPI General Linux Part 1 : Certification Study Guide
CramSession.com; ISBN: B000079Y0V; (August 17, 2000)

Referencias Unix Reviews
http://www.unixreview.com/documents/s=7459/uni1038932969999/

Página LPI: www.lpi.org

Apuntes IBM: http://www-106.ibm.com/developerworks/edu/l-dw-linux-lpir21-i.html

Manuales GPL: http://www.nongnu.org/lpi-manuals/