

Exploramos el administrador de paquetes yum

DELICIOSO

Instalación y administración de paquetes desde la línea de comandos con el potente gestor de paquetes yum. **POR DANIEL NIEDZIELSKI**

Yum (Yellow Dog Updater, Modified) se ejecuta en Fedora y Yellow Dog Linux como una interfaz al sistema RPM subyacente. Podemos pensar en él como algo parecido al administrador de paquetes apt que se encuentra en sistemas basados en Debian. Si eres partidario de una interfaz gráfica, los front-ends gráficos de yum incluyen pirut (Figura 1), gym, el extensor yum (Figura 2) y kyum; sin embargo, si estás familiarizado con unos cuantos comandos de yum básicos, probablemente encuentres que es más rápido y fácil sin la GUI. En este artículo te ayudamos a iniciarte en la línea de comandos Yum.

El administrador de paquetes yum es realmente fácil de usar. Para instalar un paquete introducimos `yum install nom-`

`bre_paquete`, pudiéndose especificar cualquier número de paquetes al mismo tiempo. Por ejemplo, para instalar el juego Pingus escribimos `yum install pingus`.

Para comprobar las dependencias, yum comienza cargando los ficheros de encabezamiento con los metadatos del paquete para el paquete nuevo como una operación en segundo plano. Tras resolver las dependencias, yum lista los requerimientos del paquete adicional (Listado 1). Seguidamente nos pregunta para confirmar que deseamos comenzar la instalación; en caso afirmativo, escribimos `Y`. Si la instalación fue exitosa, Yum nos presentará el mensaje *Complete!* para confirmar que los paquetes se instalaron correctamente.

Si no nos gusta responder *Yes/No*, podemos especificar la opción `-y` para contestar “Si” automáticamente a todas las preguntas:

```
yum -y install pingus
```

Si estamos eliminando paquetes, debemos ser cuidadosos; eliminar dependencias de manera automática puede tener efectos no deseados.

Por defecto, yum se instalará desde las fuentes de paquetes configuradas (por ejemplo, DVD, repositorios de Internet). También podemos instalar ficheros RPM individuales con `yum` en vez de con `rpm`. De nuevo, yum verificará las dependencias y consultará los repositorios para añadir los paquetes que le hacen falta automáticamente. La línea de comandos para esto es:

```
yum localinstall nombre_paquete
```

Yum también nos lo pone más fácil si necesitamos eliminar un paquete borrando cualquier componente que no vamos a necesitar más. Con el ejemplo de la instalación del juego Pingus, yum añade `sdl` y `ClanLib`.

Si deseamos eliminar estos paquetes, debemos decirle explícitamente al comando RPM que lo haga:



Elena Elisseeva, Fotolia

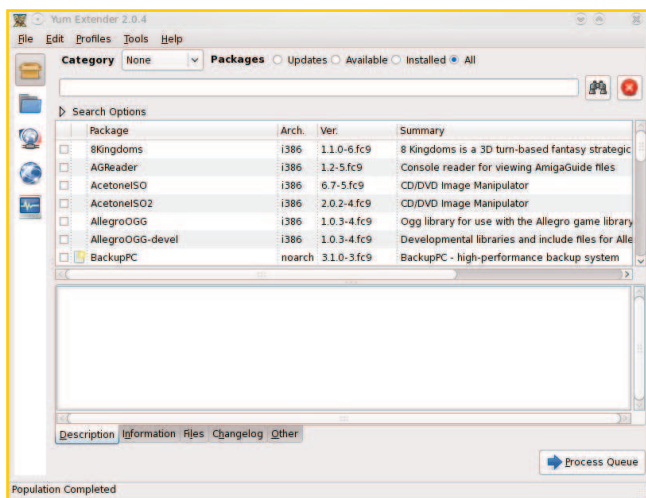


Figura 2: Si pirut es demasiado espartano para nuestro gusto, probemos con el extensor yum.

```
rpm -e pingus SDL Clanlib ...
```

Como habitualmente nos enteramos de las dependencias para los paquetes que estamos eliminando por los mensajes RPM de error, este método puede llevar bastante tiempo. En contraposición, `yum remove ClanLib` listará los paquetes a desinstalar, incluyendo sus dependencias. Yum nos pregunta rápidamente si deseamos desinstalar todos los paquetes listados, haciéndolo en caso de que contestemos afirmativamente.

Actualizar Paquetes

Para actualizar un paquete – o todos – usamos el comando `yum update nombre_paquete` o `yum update`. Si se prefiere una GUI, lanzamos el Actualizador de Paquetes, `pup`, desde una consola a través del menú escritorio. Pup forma parte del paquete `pirut`, que instalamos con `yum install pirut`.

El demonio de actualización de yum actualiza incluso el sistema completo. Para permitir que esto ocurra necesitamos configurar las entradas `do_update` a `yes` en nuestro fichero de configuración `/etc/yum/yum_updatesd.conf` (Listado 2). En el momento del arranque, `chkconfig` nos dice si el servicio `updatesd` está iniciado automáticamente:

```
# chkconfig - -list |>
grep -i yum
```

La salida muestra el nivel de ejecución en el que está habilitado el servicio. Podemos introducir `runlevel` para identi-

ficar el nivel de ejecución actual. Para habilitar el start-up automático cuando arranquemos escribimos `chkconfig yum-updatesd on`. Para iniciar el demonio una sola vez, `service yum-updatesd start`.

Probablemente usaremos muy a menudo el comando `yum search palabra_clave`, que busca los nombres de los paquetes y los detalles de todos los

paquetes disponibles según la palabra clave. Para restringir la búsqueda a los nombres solamente, podemos escribir `yum list | grep palabra_clave`.

Shell de Yum

Si necesitamos completar un par de tareas una detrás de otra, la shell yum es el mejor lugar para hacerlo. Nos permite introducir un número de acciones en secuencia sin tener que esperar a que se complete la acción previa. Un escenario típico es una sesión en la que borramos la caché, instalamos paquetes y eliminamos otros paquetes.

Para arrancar la shell escribimos `yum shell`. El prompt acepta todos los comandos normales, tales como `install`, `remove` o `update`. Con `run` iniciamos la secuencia batch que introdujimos previamente. Una vez que lo hagamos, podemos abandonar la shell introduciendo `quit`. Otro uso de la shell de yum es para invocar una tubería para instalar algo directamente desde un script de la shell:

```
# echo -e
"install
pingus\n >
run\n quit" |
yum -y shell
```

Here documents suele usarse para este propósito. Un “here document” es un método especial

para redireccionar comandos al canal `stdin` de un programa (Listado 3). A diferencia del comando `echo` normal, este método mantiene la mayoría del sangrado y de los saltos de línea.

Necesitaremos descargar algunos paquetes RPM que tendremos que instalar después, o que necesitamos instalar en algunas máquinas idénticas después. La herramienta `yum-downloader` del paquete `yum-utils` es útil para esto. Para descargar el juego Pingus introducimos `yumdownloader pingus`. Yumdownloader dejará el paquete especificado en el directorio de trabajo actual. Luego podemos introducir:

```
# yum localinstall pingus- >
0.7.2-3.fc10.i386.rpm
```

para instalar y resolver las dependencias. Si deseamos resolverlas en la fase de descarga, invocamos a `yuminstall` con la opción `--resolve`.

Yum también es perfecto para descargar paquetes fuente que compilamos manualmente. Para hacerlo usamos la opción `--source`:

```
# yumdownloader
--source pingus
```

que creará un fichero RPM llamado `pingus-0.7.2-3.fc10.i386.rpm` en el directorio actual. Para instalarlo escribimos:

```
# rpm -ihv pingus-0.7.2.3.>
fc10.i386.rpm
```

Este comando deja el archivo fuente en `/usr/src/redhat/SOURCES/`, donde podemos desempaquetar y compilar la aplicación nosotros mismos.



Figura 1: Fedora utiliza el front-end pirut como administrador de paquetes predeterminado.

Almacenamiento Temporal

Cuando arrancamos, yum actualiza las líneas de encabezado; siempre descarga los encabezamientos, incluso si estamos instalando los paquetes múltiples veces. Para evitar redundancias podemos habilitar el encabezado y el caching del fichero de paquetes. Para hacerlo, habilitamos la opción `keepcache=1` en el fichero de configuración `/etc/yum.conf`. Si luego llamamos a `yum` con la opción `-C`, yum accederá a la caché a pesar de las configuraciones `/etc/yum.conf`. Evidentemente, esto sólo funcionará si realmente hacemos que estén los recursos requeridos en la caché. Si no se encuentran, yum nos dará el mensaje `Caching enabled but no local cache of`

Plugins y Repositorios

Yum posee muchas extensiones que podemos instalar fácilmente desde el repositorio. Cuando descargamos un fichero, por ejemplo, el plugin `Fastest-mirror` realiza una búsqueda en segundo plano para el servidor espejo más rápido. Para instalar y habilitar el plugin introducimos `yum install yum-fastestmirror`.

Si existen varios mirrors disponibles, se instala el plugin `Priorities`, el cual nos permite controlar las prioridades de los repositorios individuales:

```
yum -y install yum-priorities
```

Después de introducir este comando, encontraremos un fichero `priorities.conf` en `/etc/yum/pluginconf.d/`:

```
ain*
enabled = 1
check_obsoletes = 1
```

Para evitar problemas legales, Fedora sólo proporciona software disponible bajo licencia libre. Los repositorios Fedora predeterminados no incluyen programas propietarios, o por razones de licencias, muchas otras aplicaciones libres, como `mplayer`. Afortunadamente, algunos de estos programas se encuentran disponibles a través de proveedores de tercer grupo.

Los repositorios más populares incluyen Livna, RPMforge, Dribble, freshrpms y ATrpms. En Fedora 10, Dribble, freshrpms, Atrpms y Livna se

combinan para formar un único repositorio conocido como RPM Fusion [1]. La página FAQ no oficial de Fedora [2] describe cómo añadir Livna, o RPM Fusion, como paquete fuente; para Fedora 10, simplemente introducimos:

```
# rpm -Uvh ↗
http://rpm.livna.org/ ↗
livna-release-10.rpm
```

Los ficheros de configuración para los diferentes repositorios se almacenan en `/etc/yum-repos.d/`. En estos ficheros de configuración, podemos habilitar o deshabilitar repositorios individuales con entradas `enabled = 1` o `enabled = 0`. La mayoría de los ficheros de repositorios incluyen varias entradas, aunque sólo necesitaremos habitualmente la superior.

Después de completar una instalación Fedora estándar, incluyendo el repositorio RPM Fusion, tendremos tres ficheros de configuración bajo `/etc/yum.repos.d/`: `fedora.repo`, `fedora-updates.repo` y `livna.repo`. Configuramos las líneas superiores a `enabled = 1` y las otras a `enabled = 0`.

Alternativamente, podemos borrar y deshabilitar todas las entradas del repositorio adicionales escribiendo:

```
# rpm -e yum-priorities ↗
yum-fedoraFAQ livna-release
```

El plugin `yum-priorities` mencionado antes se usa aquí. `yum repolist` ofrece una lista de todos los repositorios habilitados.

Actualizando con Yum

Yum permite actualizar al vuelo un sistema Fedora a la última versión. Desafortunadamente, esta opción a veces no funciona de forma adecuada y requiere habitualmente alguna atención manual.

Si decidimos confiar en la actualización de yum, debemos comenzar borrando nuestra caché yum y actualizando las fuentes de paquetes. El Listado 4 ofrece una escueta perspectiva de los pasos requeridos para actualizar a Fedora 11. Encontraremos un howto más detallado en la wiki de Fedora [3].

Grupos de Paquetes

Si usamos Ubuntu y de vez en cuando necesitamos compilar algo, deberemos estar familiarizados con el paquete `build-essential`. Este paquete no es un paquete Debian en sentido estricto, sino un meta-paquete que instala automáticamente una selección de software. Fedora dispone de una funcionalidad similar aunque la llama `grupos de paquetes`.

`yum grouplist` nos da una lista de los grupos de paquetes disponibles. Nótese

Listado 1: Instalando con Yum

```
Dependencies Resolved
=====
Package      Arch      Version                Repository             Size
=====
Installing:
pingus       i386      0.7.0-0.4.20060721.fc6 fedora                 18 M
Installing for dependencies:
ClanLib      i386      0.8.0-4.fc7           fedora                 1.2 M
SDL          i386      1.2.12-1.fc7          updates                219 k
SDL_gfx      i386      2.0.13-8.fc7          fedora                 52 k
...
Transaction Summary
=====
Install     11 Package(s)
Update      0 Package(s)
Remove      0 Package(s)
Total download size: 31 M
Is this ok [y/N]:
...
Complete!
```

Listado 2: yum-updatesd.conf

```
[main]
# frecuencia de comprobación de actualizaciones (en segundos)
run_interval = 3600
# con frecuencia se permiten comprobaciones a petición de los usuarios
(en segundos)
updaterefresh = 600

# como enviar notificaciones (opciones válidas: dbus, email, syslog)
emit_via = dbus

# instalar automáticamente actualizaciones
do_update = yes
# descargar automáticamente actualizaciones
do_download = no
# descargar automáticamente dependencias de actualizaciones
do_download_deps = no
```

el grupo *Development Tools*, que podemos instalar escribiendo:

```
# yum -y groupinstall >
'Development Tools'
```

Si deseamos instalar todos los juegos disponibles, o quizás todos los programas oficiales, seleccionamos la categoría deseada de la lista e instalamos como se ha descrito anteriormente.

Función Rollback

Probablemente nos encontremos familiarizados con la situación en la que instalamos un nuevo programa aunque después necesitamos restaurar el estado original del sistema debido a incompatibilidades, errores u otros problemas. Esta desinstalación puede ser bastante difícil. Por un lado, los paquetes más antiguos pueden ser difíciles de localizar, y por otro lado, las configuraciones pueden haber sido sobrescritas. Una función rollback automática es a menudo el método más rápido, simple y limpio.

Para mejorar los beneficios del paquete rollback necesitamos hacer

copias de seguridad del estado del sistema actual antes de cambiar la selección de paquetes – es decir, siempre que instalemos o borremos paquetes. Para ayudarnos a hacerlo, RPM dispone de una función de repaquetado.

Este proceso de repaquetado maneja ficheros importantes, tales como los de configuración, datos y ficheros de programas, en un nuevo archivo RPM. Esta función de repaquetado RPM añade una ID de transacción para cada archivo y guarda cada archivo en un sitio definido de antemano.

Para restaurar un estado de instalación previo, eliminamos los paquetes que instalamos e instalamos los ficheros repaquetados que creamos con anterioridad. Para decirle a RPM qué ficheros repaquetados instalar, cada transacción RPM está asignada a un único número. Si eliminamos múltiples paquetes y sus dependencias, todos los ficheros repaquetados tendrán una única ID de transacción. Según el manual de RPM, la ID es una marca de tiempo Unix simple.

Yum puede usar funcionalidad rollback de RPM si la habilitamos en los ficheros de configuración. Para hacerlo,

añadimos la entrada siguiente a */etc/rpm/macros*:

```
%_repacke_all_erasures 1
```

Si el fichero de configuración */etc/rpm/macros/* no existe, necesitamos crearlo. A continuación añadimos la línea siguiente a */etc/yum.conf*:

```
tsflags=repackage
```

Una vez que hemos configurado este parámetro, el sistema “recordará” cada estado configurado en la carpeta */var/spool/repackage*. Si borramos un paquete, algo como

```
# rpm -Uhv - -rollback >
'3 minutes ago'
```

restaurará el estado del sistema de hace tres minutos.

Pruebas

Adicionalmente, especificando *--test* se llevará a cabo un test que nos mostrará las acciones a realizar, aunque sin que de hecho se ejecute el rollback. Podemos especificar la hora del rollback como sigue:

```
# rpm -Uhv - -rollback >
'YYYY-MM-DD HH:MM'
```

Si no podemos recordar la hora de instalación que queremos recuperar, verificamos los sellos de tiempo en los ficheros de repaquetado en */var/spool/repackage*.

Listado 3: Here Document

```
#yum -y shell <<EOF
clean all
install ...
remove ...
run
quit
EOF
```

RECURSOS

- [1] RPM Fusion: <http://rpmfusion.org>
- [2] Fuentes de paquetes adicionales: <http://www.fedorafaq.org/#getsoftware>
- [3] Yum Upgrade: <http://fedoraproject.org/wiki/YumUpgradeFaq>

Listado 4: Actualizando a Fedora 11

```
yum clean all
rpm -Uhv ftp://download.fedora.redhat.com/pub/fedora/linux/releases/11/
Fedora/i386/os/Packages/fedora-release-*.noarch.rpm
yum upgrade
```