



123RF (Exclusive), 123 RF

Yummy, yummy, yummy

PAQUETES SABROSOS

Yum, el administrador de paquetes de RPM, tiene sus propias ventajas sobre otras herramientas.

POR BRUCE BYFIELD

Hace una década, los usuarios de Debian solían mirar con cierto desprecio a los usuarios de Red Hat y de otras distribuciones basadas en RPM. Mientras que los de Debian tenían apt-get y dpkg para salvarles del demonio de las dependencias – situaciones en las que la instalación de software es imposible debido a una librería no instalada – los usuarios de distribuciones basadas en RPM debían quitárselo de encima ellos mismos.

Hoy, los usuarios de Debian y Ubuntu no tienen ninguna razón para ser tan engreídos. Las distros de RPM se han puesto al nivel de las herramientas

Debian y producido algunos administradores de paquetes que igualan más o menos la funcionalidad de apt-get.

De dichos administradores de paquetes, el más popular es *Yellowdog Updater, Modified*, más conocido como Yum. Esta reescritura de un temprano administrador de paquetes para Yellow Dog Linux se mantiene actualmente a través de Seth Vidal, un empleado de Red Hat, y es usado por muchas de las principales distribuciones RPM, incluyendo Fedora, Red Hat y CentOS.

Al igual que apt-get en Debian proporciona a los usuarios acceso a la funcionalidad de *dpkg*, del mismo modo Yum

actúa como un encapsulador para *rpm*, el comando básico para la administración de paquetes RPM. La principal diferencia es que, mientras que *dpkg* resuelve problemas de dependencias por sí mismo, *rpm* no. Esa funcionalidad reside completamente en Yum.

Desde la perspectiva del usuario, esta diferencia no tiene importancia. A pesar de que podemos utilizar Yum indirectamente desde cualquier aplicación de escritorio como PackageKit y Yumex, Yum está tan bien escrita que podemos aprender a ejecutarla de manera fácil directamente desde la línea de comandos.

```
[root@localhost ~]# yum install sil-gentium-basic-book-fonts.noarch
Loaded plugins: dellsysidplugin2, presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package sil-gentium-basic-book-fonts.noarch 0:1.1-4.fc11 set to be updated
--> Processing Dependency: sil-gentium-basic-fonts-common = 1.1-4.fc11 for packa
ge: sil-gentium-basic-book-fonts-1.1-4.fc11.noarch
--> Running transaction check
--> Package sil-gentium-basic-fonts-common.noarch 0:1.1-4.fc11 set to be update
d
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch      Version      Repository  Size
=====
Installing:
sil-gentium-basic-book-fonts      noarch    1.1-4.fc11   fedora      425 k
Installing for dependencies:
sil-gentium-basic-fonts-common    noarch    1.1-4.fc11   fedora      21 k
=====

Transaction Summary
=====
Install      2 Package(s)
Update      0 Package(s)
Remove      0 Package(s)

Total download size: 446 k
Is this ok [y/N]: █
```

Figura 1: Yum nos da mensajes constantes sobre lo que está haciendo.

Aprendemos los Básicos

Yum, como apt-get, posee un formato básico consistente: el comando básico (*yum*), el subcomando (lo que queremos hacer) y los paquetes involucrados. La diferencia principal está en la lista de subcomandos que intervienen. Yum es más organizado que apt-get, y algunos de sus subcomandos proporcionan funcionalidades que en apt-get residen en una utilidad relacionada, pero separada, tal como apt-cache.

El subcomando que probablemente usaremos más será *install*. Por ejemplo, si planeamos instalar el tipo de letra Book para la fuente libre Gentium, el comando básico en Fedora sería:

```
yum install ?
sil-gentium-basic-book-fonts
```

como sería con apt-get (aunque el nombre exacto del paquete podría diferir).

Sin embargo, Yum es un tanto más prolijo que apt-get a la hora ofrecer feedback (Figura 1). Comienza listando los plugins que están instalados para Yum, luego calcula qué paquetes necesitan ser actualizados y las dependencias que necesita el paquete requerido. Por ejemplo, si decidimos instalar Gentium Book, Yum apuntará que requiere el paquete *sil-gentium-basic-fonts-common*, el cual es necesario con cualquier peso de Gentium que instalemos.

Finalmente, Yum observa si se encuentran disponibles todas las dependencias y presenta todo lo que necesita ser instalado en una tabla. A ésta le sigue una segunda que lista las transacciones (por ejemplo, pasos) necesarias para completar la instalación y la cantidad total de

espacio de disco requerido. Solamente entonces nos ofrece la opción de continuar o detener la instalación.

Una vez que pulsamos y (sí) para continuar el proceso de instalación, Yum comienza a descargar los paquetes necesarios, mostrando el progreso de cada descarga y del total

de procesos. Después de que las descargas se han completado, instala cada paquete y resume lo que ha hecho (un paso útil, en el que la información general podría fácilmente haberse desplazado fuera de la vista). Si tiene éxito, presenta un escueto *Complete!* antes de salir (Figura 2).

Con apt-get usaríamos dpkg para instalar un paquete que habríamos descargado a un sistema Debian; sin embargo, con Yum simplemente usaríamos el subcomando *localinstall* y no tendríamos que cambiar a otro comando básico.

Para instalar una versión más nueva de un paquete también podemos utilizar *install*, aunque es mejor elección el subcomando *upgrade* porque puede controlar la eliminación de cualquier paquete obsoleto – una funcionalidad que es especialmente útil cuando estamos cambiando de una versión de distribución a otra. Para obtener el mismo resultado podemos usar *yum update* —*obsoletes*. Si somos cautelosos, puede que prefiramos el subcomando *check-update* para ver que está disponible antes incluso de instalar nada. O, en su lugar, puede que prefiramos especificar paquetes individuales para su actualización.

Para instalar nuevas versiones de paquetes locales, el

subcomando es *localupdate*. Para desinstalar empleamos *remove*.

Todos estos subcomandos básicos se encuentran disponibles para su uso con múltiples paquetes. La manera más fácil de manejar estos últimos es introduciendo un espacio entre ellos al final del comando. Alternativamente, podemos usar expresiones regulares, aunque lo mejor es emplear el subcomando *search* primero para ver qué paquetes estarán afectados.

Algunos repositorios Yum organizan paquetes en grupos. Por ejemplo, en Fedora 11, los grupos de paquetes incluyen Games, KDE Desktop y Publishing. Estos grupos hacen la misma función que los meta-paquetes en sistemas Debian, permitiéndonos instalar múltiples paquetes sin tener que recordarlos o editarlos separadamente. Los grupos poseen una serie de subcomandos especiales que incluyen *groupinstall*, *groupupdate* y *groupremove*, seguido por el nombre del grupo. Por ejemplo, *groupinstall publishing* añadiría todos los ficheros a nuestro sistema en el grupo publishing.

Subcomandos de Información

Además de estos subcomandos básicos, Yum incluye algunos otros que proporcionan información o nos ayudan a mantener nuestro sistema.

El subcomando más básico, *list*, se completa con descripciones auto-explicativas de la información que deseamos. Por ejemplo, el comando *yum list installed* muestra una lista completa de paquetes instalados. De forma similar,

```
root@localhost ~
File Edit View Terminal Help
Total download size: 2.5 M
Is this ok [y/N]: y
Downloading Packages:
Setting up and reading Presto delta metadata
fedora/prestodelta | 410 B 00:00
Processing delta metadata
Package(s) data still to download: 2.5 M
(1/2): PySolFC-music-4.40-4.noarch.rpm | 1.7 MB 00:02
(2/2): pygame-1.8.1-6.fc11.i586.rpm | 882 kB 00:01
-----
Total | 603 kB/s | 2.5 MB 00:04
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
Installing : pygame-1.8.1-6.fc11.i586 | 1/2
Installing : PySolFC-music-4.40-4.noarch | 2/2

Installed:
PySolFC-music.noarch 0:4.40-4

Dependency Installed:
pygame.i586 0:1.8.1-6.fc11

Complete!
```

Figura 2: Cuando instalamos los paquetes, Yum nos mantiene informados de su progreso. Nótese que está verificando DeltaRPMs, lo que indica que Yum está utilizando el plugin yum-presto para minimizar el tamaño de las descargas.

```
[root@localhost ~]# yum info evolution
Loaded plugins: dellsysidplugin2, presto, refresh-packagekit
Installed Packages
Name      : evolution
Arch      : i586
Version   : 2.26.3
Release   : 1.fc11
Size      : 38 M
Repo      : installed
Summary   : Mail and calendar client for GNOME
URL       : http://projects.gnome.org/evolution/
License   : GPLv2+ and GFDL
Description: Evolution is the GNOME mailer, calendar, contact manager and
           : communications tool. The components which make up Evolution
           : are tightly integrated with one another and act as a seamless
           : personal information-management tool.
```

Figura 3: El subcomando "info" proporciona información disponible sobre paquetes.

yum list available lista todos los paquetes en todos los repositorios y *yum list updates*, todas las actualizaciones disponibles. Otro descripción, más específica, incluye *extras*, el cual lista paquetes en nuestro sistema que no están listados en repositorios no habilitados, y *recent*, que lista las últimas actualizaciones en los repositorios.

Cuando deseamos información más específica sobre un paquete, el subcomando a usar es *info*, seguido por el nombre del paquete. El comando *info* proporciona información básica sobre el paquete: su arquitectura, número de versión y edición; si está instalado o, si no, en qué repositorio está; su licencia y su página de inicio (Figura 3). También recibiremos un resumen de una única frase y una descripción ligeramente larga. El subcomando *groupinfo* ofrece información similar para grupos de paquetes.

Uno excepcional, aunque ocasionalmente útil, es *provides*. Con el comando *provides* podemos encontrar qué paquete incluye un fichero o funcionalidad específica (Figura 4). Por ejemplo, en Fedora 11, el comando *yum list provides firefox* devuelve exactamente qué versión de paquete está disponible o instalada además de la versión encontrada en los repositorios.

Otro medio de rastreo de referencias a un paquete específico es el subcomando *search*. Esta función localizará todos los paquetes y dependencias relacionadas con el término de búsqueda, seguido de una breve descripción. Similar a *apt-cache* en sistemas Debian, *search* puede ser útil para encontrar paquetes cuando carecemos de un nombre exacto o estamos razonablemente seguros de que una función debe estar disponible en algún sitio.

Todos estos subcomandos de información ofrecen frecuentemente docenas, incluso cientos de líneas de salida. Por

esta razón, debemos considerar redirigirlos a través del comando *less* de manera que podamos desplazarnos a nuestro placer. Por ejemplo, con *yum list obsoletes | less* podemos mirar una lista de los paquetes instalados que se han vuelto obsoletos por paquetes en los repositorios.

Subcomandos de Mantenimiento

Los subcomandos de Yum incluyen también una serie de utilidades para ayudarnos a mantener y resolver problemas de nuestro sistema. Por ejemplo, *yum makecache* descarga la información para todos los paquetes en todos los repositorios habilitados, los cuales podemos usar si la información está corrupta o si hemos cambiado repositorios recientemente. Asimismo, en el momento excepcional en el que emergen de repente los problemas de dependencia, *yum resolvedep* puede decirnos qué paquetes proporcionan una dependencia faltante.

Una herramienta de mantenimiento particularmente potente es *clean*, que, al igual que *list*, está completada por una descripción de la fuente de información que deseamos eliminar. Sin embargo, con la excepción del comando *yum clean packages*, que elimina paquetes que fueron descargados pero no instalados, el uso de *clean* es un acto de desesperación.

Ejecutar *clean* seguido de cualquier otra opción - *expire-cache*, *headers*, *meta-data*, *dbcache* o *all* - elimina información que Yum necesita para operar. La próxima vez que iniciemos Yum después de ejecutar *clean* con estas terminaciones, Yum volverá a crear lo que se borró, aunque podría llevarle hasta 20 minutos hacerlo, dependiendo de la máquina. Por esta razón, sólo deberíamos ejecutar el subcomando *clean* cuando estamos teniendo problemas con Yum y han fallado todos los demás medios de resolución de problemas. A diferencia de *clean* y *autoclean* de *apt-get*, *clean* de Yum no es para un mantenimiento de rutina, sino para problemas serios, y sólo nos causaremos molestias a nosotros mismos si lo ejecutamos de manera informal.



Dirigido a profesionales y usuarios de las TIC

CURSOS GRATUITOS
SOBRE

SOFTWARE LIBRE

<http://www.morfeo-formacion.org>



Introducción al software libre



Gestión de proyectos de software libre



Migración al software libre



Introducción a escritorios GNU/Linux: Molinux



Morfeo EzWeb



MyMobileWeb

<http://www.morfeo-formacion.org>



Síguenos en las redes sociales:



Opciones

La mayoría de las veces podemos usar Yum sin opciones. Algunas, como `--obsoletes`, suministran información útil para ayudarnos a administrar la instalación de software. Una gran mayoría de veces, sin embargo, habilita o deshabilita información para varios propósitos.

Algunas opciones, tales como `-d` y `-e`, las cuales establecen informes de nivel de error y depurado, son en gran parte para desarrolladores. Lo mismo ocurre con `-v` o `--verbose`, dos opciones equivalentes que aumentan la cantidad de información que Yum suministra mientras se ejecuta.

Otras opciones son para usuarios que desean utilizar Yum con un mínimo de escándalo, como `--quiet`, que hace que Yum se ejecute sin informar de lo que está haciendo. Un compañero frecuente es `-y`, el cual supone que las respuestas a todas las preguntas es "Sí" – incluyendo la pregunta de si deseamos continuar después de que Yum finalice sus cálculos iniciales. De la misma forma, `--nogpgcheck` deshabilita la verificación del paquete.

Tales opciones también nos ahorran tiempo y molestias. Sin embargo, sugiero que se eviten según el principio general de que abandonar el control es raras veces una buena idea. Algo tan simple como un error de mecanografía podría hacer que Yum comenzara una serie de acciones que podrían destrozar nuestro sistema – o al menos que precisara algunas reparaciones.

Otras opciones es menos probable que causen problemas. La pareja `--enable-repo=` y `--disable-repo=` dan a Yum el equivalente del direccionamiento de apt-get y especifican qué repositorios utilizar. También podemos usar `--exclude=` para evitar paquetes que podrían causar un conflicto de instalación desde cualquier fuente.

Otra opción que tampoco puede evitarnos problemas es `--skip-broken`. Si la usamos después de que Yum nos informe sobre la pérdida de una dependencia, debería permitirnos resolver la dificultad. En algunos casos, el paquete instalado con esta opción no funcionará, pero podemos asegurarnos de que no forma un cuello de botella que impide que Yum trabaje. Una vez que se han instalado, podemos eliminarlos con normalidad.

Plugins

Los comandos de Yum y las opciones proporcionan toda la funcionalidad que necesitan la mayoría de los usuarios. Sin embargo, si estamos buscando algo extra, o si deseamos ver lo que puede ser el futuro de Yum, tomémonos un momento para examinar los plugins disponibles para nuestra distribución. Escritos en Python, los plugins de Yum se encuentran disponibles como paquetes separados en ella. Con cada plugin añadimos nuevas funcionalidades.

Por defecto, no se encuentran activados. Antes de poder usar ninguno de ellos, necesitamos abrir `/etc/yum.conf` y añadir o editar la línea `plugins` en la sección principal del fichero para que diga `plugins = 1`.

En Fedora y Red Hat, algunos de los plugins más comunes se encuentran disponibles en el paquete yum-utils. Actualmente también se encuentran disponibles otros 20. Sus funcionalidades cubren casi todo lo que podamos imaginar, siendo demasiados para describirlos aquí detalladamente.

Brevemente, sin embargo, algunos plugins útiles en Fedora 11 son:

- `yum-plugin-version-lock`: Evita que una versión particular del programa se sobrescriba.
- `yum-plugin-protects-packages`: Evita que sean eliminados paquetes designados (incluyendo Yum).
- `yum-plugin-allowdowngrades`: Nos permite bajar de versión un paquete – una operación que es a veces necesaria para compatibilidad entre paquetes.
- `yum-plugin-fastest-mirror`: Lista los mirrors más rápidos para los repositorios que pedimos.
- `yum-presto`: Ordena a Yum que busque DeltaRPMs en vez de RPMs. Los DeltaRPMs son paquetes que incluyen solamente cambios en un paquete, de este modo, usándolos, podemos instalar un paquete más rápidamente y con menos ancho de banda.
- `yum-plugin-security`: Añade opciones para limitar las actualizaciones a aquellos que sean por seguridad.

Muchos plugins, si no todos, funcionan automáticamente, por lo que no incluyen ni páginas man ni de ayuda. Sin embargo, podemos ver los que se

```
firefox-3.5.2-2.fc11.i586 : Mozilla Firefox Web browser
Repo                    : installed
Matched from:
Filename                : /usr/bin/firefox
Filename                : /usr/lib/firefox-3.5.2/firefox
```

Figura 4: Si nos preguntamos de dónde vienen los ficheros o aplicaciones, "yum list provides" nos puede facilitar esta información. Nos da ambas fuentes, las actuales y las posibles.

encuentran instalados en nuestro sistema con el comando `search yum`.

A veces podemos encontrar que un plugin, o quizás un conflicto entre dos o más plugins, impide que Yum funcione adecuadamente. Si esto ocurriera, podemos usar la opción `--disableplugin=[plugin name]` para ayudarnos a resolverlo deshabilitando un plugin específico. Si nos encontramos en apuros, podemos usar la opción `--noplugins` para ejecutar Yum sin plugins de manera que, con suerte, podamos recuperar el uso del programa.

Otros Administradores de Paquetes

Yum no es el único administrador de paquetes basado en RPM disponible. Mandriva usa una herramienta similar llamada urpmi y OpenSUSE utiliza Smart, un administrador que no sólo se parece a Yum, sino que podemos instalar desde los repositorios de Yum. Otros sistemas de administración de paquetes incluyen Portage and Conary de Gentoo, el cual incluye un control de versión que nos permite instalar diferentes versiones de los mismos paquetes.

Sin embargo, Yum es casi con toda seguridad el administrador de paquetes más comúnmente usado después de apt-get de Debian y Ubuntu. Además, como fue desarrollado años después que apt-get, sus desarrolladores tuvieron la oportunidad de aprender de apt-get y de mejorarlo.

A pesar de que probablemente no veremos mucho en Yum que no esté en apt-get y en sus utilidades relacionadas, sin embargo encontraremos una organización menos caótica. De hecho, si conocemos apt-get y estamos encontrándonos por primera vez con Yum, probablemente quedaremos sorprendidos por cómo está estructurado Yum y lo fácil que es aprenderlo si lo comparamos con apt-get. Como programa, Yum es una innovación inteligente – y como remedio para el demonio de las dependencias, una bendición. ■