



123RF (Exclusive) 123RF

Yummy, yummy, yummy

SERVE ME RIGHT

The RPM package manager Yum has its own advantages over other tools. **BY BRUCE BYFIELD**

A decade ago, Debian users used to look down on users of Red Hat and other RPM-based distributions. Whereas Debian users had apt-get and dpkg to save them from dependency hell – situations in which software installation is impossible because of an uninstalled library – users of RPM-based distributions had to dig themselves out of dependency hell on their own.

Today, Debian and Ubuntu users have no reason to be so smug. The RPM distros have caught up with the Debian tools and produced several package managers that more or less equal apt-get's functionality.

Of these package managers, the most popular is Yellowdog Updater, Modified,

better known as Yum. This rewriting of an earlier package manager for Yellow Dog Linux is now maintained by Seth Vidal, a Red Hat Employee, and used by many of the major RPM distributions, including Fedora, Red Hat, and CentOS.

Just as apt-get in Debian provides users access to the functionality of *dpkg*, so Yum acts as a wrapper for *rpm*, the basic command for RPM package management. The main difference is that, whereas *dpkg* resolves dependency problems on its own, *rpm* does not. That functionality resides entirely in Yum.

From the user's perspective, this difference is unimportant. Although you can use Yum indirectly from such desktop applications as PackageKit and

Yumex, Yum is so well written that you can easily learn to run it directly from the command line.

Learning the Basics

Like apt-get, Yum has a consistent basic format: the basic command (*yum*), the sub-command (what you are doing), and the packages involved. The main difference is in the list of sub-commands involved. Yum is more organized than apt-get, and some of its sub-commands provide functionality that with apt-get reside in a related utility, such as apt-cache.

The sub-command that you will probably use most often is *install*. For instance, if you plan to install the Book

```
[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]: █
```

Figure 1: Yum gives you constant messages about exactly what it is doing.

typeface for the free Gentium font, the basic command in Fedora would be

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

much as it would be with apt-get (although the exact package name might differ).

However, Yum is somewhat more verbose than apt-get in offering feedback (Figure 1). It begins by listing which plugins are installed for Yum, then it calculates which packages need to be updated and which dependencies the requested package needs. For instance, if you decided to install Gentium Book, Yum would note that it requires the package *sil-gentium-basic-fonts-common*, which is needed with any weight of Gentium that you install.

Finally, Yum notes whether all dependencies are available and presents everything that needs to be installed in a table. This table is followed by a second that lists the transactions (i.e., steps) needed to complete the installation and the total amount of hard disk space required. Only then does Yum offer you a choice of whether to continue or halt installation.

Once you press y (for “yes”) to continue the installation process, Yum begins to download the necessary packages, showing the progress of each download and of the overall process. After the downloads are complete, Yum

installs each package and summarizes what it has done (a useful step, in that the original information might easily have scrolled out of sight). If it is successful, a succinct *Complete!* displays just before Yum exits (Figure 2).

With apt-get, you would use dpkg to install a package you had downloaded to a Debian system; however, with Yum you would simply use the *localinstall* sub-command and not have to switch basic commands.

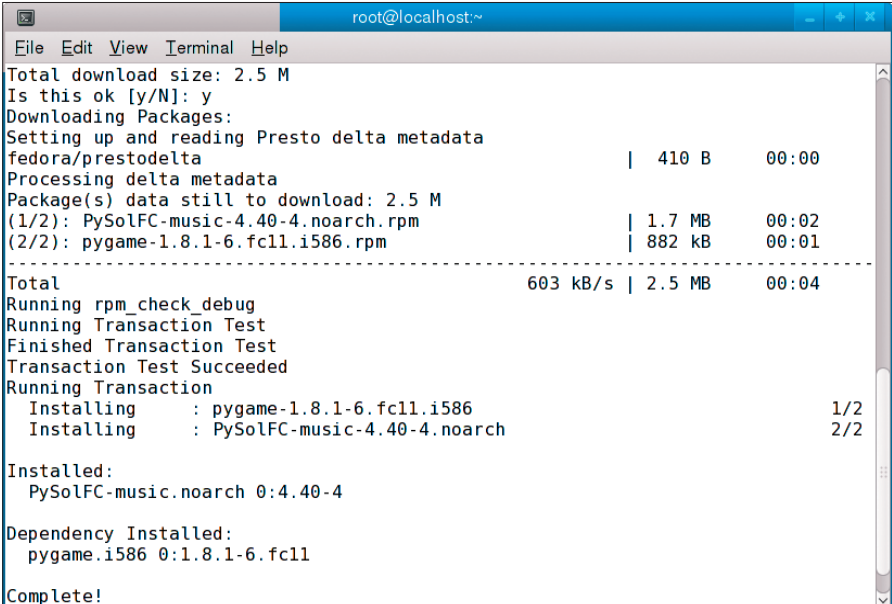
To install a newer version of a package, you can also use install, but a better

choice is the *upgrade* sub-command because it can handle the removal of any obsolete packages – an ability that is especially useful when you are switching from one version of a distribution to another. For the same result, you can use *yum update --obsoletes*. If you are cautious, you might prefer to use the *check-update* sub-command to see what is available before actually installing anything. Or, you might prefer to specify particular packages to upgrade instead.

To upgrade local packages, the sub-command is *localupdate*. To uninstall, use the *remove* sub-command.

All of these basic sub-commands are available for use on multiple packages. The simplest way to handle multiple packages is to enter a space between them at the end of the command. Alternatively, you might want to use regular expressions, although generally you should use the *search* sub-command first to see which packages will be affected.

Some Yum repositories organize packages into groups. For example, in Fedora 11, the package groups include Games, KDE Desktop, and Publishing. These groups serve much the same function as meta-packages on Debian systems, allowing you to install multiple packages without having to remember them or edit them separately. Groups have a series of special sub-commands that include *groupinstall*, *groupupdate*, and *groupremove*, followed by the name of



```
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!
```

Figure 2: When installing packages, Yum keeps you informed of its progress. Note that it is checking for DeltaRPMs, which indicates that Yum is using the yum-presto plugin to minimize the size of downloads.

```
[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.
```

Figure 3: The "info" sub-command gives you all available information about packages.

the group. For example, `yum groupinstall publishing` would add all the files in the publishing group to your system.

Information Sub-Commands

Besides to these basic sub-commands, Yum also includes several that provide information or help you to maintain your system.

The most basic sub-command, `list`, is completed by self-explanatory descriptions of the information you want. For instance, the command `yum list installed` displays a complete list of installed packages. Similarly, `yum list available` lists all the packages in all repositories, and `yum list updates` lists all available updates. Other, more specific, descriptions include `extras`, which lists packages on your system that are not listed in enabled repositories, and `recent`, which lists the latest updates in the repositories.

When you want more specific information about a package, the sub-com-

mand to use is `info`, followed by the package name. The `info` command provides basic information about the package: its architecture; its version number and release; whether it is installed or, if not, what repository it is in; its license; and its homepage (Figure 3). Also, you will receive a single-sentence summary and a slightly longer description. The sub-command `groupinfo` provides similar information for package groups.

A rarer, but occasionally useful, sub-command is `provides`. With the `provides` command, you can find which package includes a particular file or feature (Figure 4). For example, in Fedora 11, the command `yum list provides firefox` returns exactly which package version is available or installed as well as the versions found in the repositories.

Another means of tracing references to a specific package is the

`search` sub-command. This function will locate all packages and dependencies related to the search term, followed by a brief description. Similar to `apt-cache` on Debian systems, `search` can be useful for finding packages when you lack an exact name or are reasonably sure that a function must be available somewhere.

All of these information sub-commands frequently give dozens, even hundred of lines of output. For this reason, consider piping them through the `less` command so that you can scroll through at your leisure. For example, with `yum list obsoletes | less`, you can browse a list of the installed packages that are made obsolete by packages in the repositories.

Maintenance Sub-Commands

Yum sub-commands also include a series of utilities to help you maintain and troubleshoot your system. For instance, `yum makecache` downloads the information for all packages in all enabled repositories, which you can use if the information is corrupted or if you have recently changed repositories. Similarly, for the rare time that dependency problems suddenly emerge, `yum resolvedep`

```
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
```

Figure 4: If you wonder where files or applications come from, "yum list provides" can give you the information. Both possible and actual sources are given.

Best Price Guarantee!

Online. Easy. Secure. Reliable
All you need to run your home business or small office:

- Accounting Software
- Business Planning
- Online Data Storage
- Business Academy
- Email/Fax/SMS
- Sales Invoicing



- Online Shop
- Calendar
- Contacts
- Payment
- Web Hosting
- Networking

can tell you which packages provide a missing dependency.

A particularly powerful maintenance tool is *clean*, which, like *list*, is completed by a description of the information source that you want to remove. However, with the exception of the command *yum clean packages*, which removes packages that were downloaded but not installed, using *clean* is an act of desperation.

Running *clean* followed by any other option – *expire-cache*, *headers*, *metadata*, *dbcache*, or *all* – removes information that Yum requires to operate. The next time you start Yum after running *clean* with these completions, Yum will rebuild what was deleted, but it could take as long as 20 minutes to do so, depending on your machine. For this reason, you should only run the *clean* subcommand when you are having trouble with Yum and all other means of troubleshooting have failed. Unlike *apt-get*'s *clean* and *autoclean*, Yum's *clean* is not for routine maintenance, but for major problems, and you will only inconvenience yourself if you run it casually.

Options

Most of the time, you can use Yum without any options. A few options, such as *--obsoletes*, provide useful information to help you administer software installation. A great many more, however, enable or disable information for various purposes.

Some options, such as *-d* and *-e*, which set debugging and error-level reporting, are largely for developers. The same is true of *-v* or *--verbose*, two equivalent options that increase the amount of information that Yum provides while running.

Other options are for users who want to use Yum with a minimum of fuss, such as *--quiet*, which causes Yum to run without reporting what it is doing. Its frequent companion is *-y*, which assumes that the answers to all questions are “Yes” – including the question of whether you want to proceed after Yum finishes its initial calculations. In much the same way, *--nogpgcheck* disables package verification.

Such options save time as well as your own watchfulness. However, I suggest that you avoid them on the general principle that giving up control is rarely a

good idea. If nothing else, something as simple as a typing error could start Yum off on a series of actions that could trash your system – or at least require some careful repairs.

Other options are less likely to cause trouble. The matched pair *--enable-repo =* and *--disable-repo =* give Yum the equivalent of *apt-get*'s pinning and specify which repositories to use. Also, you can use *--exclude =* to prevent packages that could cause a conflict from installing from any source.

Another option that might keep you out of trouble is *--skip-broken*. If you use it after Yum reports a missing dependency, it might just allow you to resolve the difficulty. In some cases, the packages installed with this option will not work, but you can make sure that they do not form a bottleneck that keeps Yum from working. Once they are installed, you can then delete them normally.

Plugins

Yum's commands and options provide all the functionality most users will need. However, if you're looking for something extra, or you want to see what the future of Yum might be, take the time to look at the plugins available for your distribution. Written in Python, Yum plugins are available as separate packages for your distribution, with each plugin adding new features.

By default, plugins are not turned on. Before you can use any of them, you need to open */etc/yum.conf* and add or edit the *plugins* line in the main section of the file to read *plugins = 1*.

In Fedora and Red Hat, some of the most common plugins are available in the *yum-utils* package. Approximately another 20 plugins are also currently available. Their functionality covers almost everything you can imagine and is too much to describe in detail here.

Briefly, though, some useful Yum plugins in Fedora 11 are:

- *yum-plugin-versionlock*: Prevents a particular version of a program from being overwritten.
- *yum-plugin-protect-packages*: Prevents designated packages (including Yum) from being removed.
- *yum-plugin-allowdowngrade*: Allows you to downgrade a package – an operation that is needed sometimes for compatibility between packages.

- *yum-plugin-fastestmirror*: Lists the fastest mirrors for the repositories you request.
 - *yum-presto*: Instructs Yum to look for DeltaRPMs rather than straight RPMs. DeltaRPMs are packages that include only changes in a package, so by using them, you can install a package faster and with less bandwidth.
 - *yum-plugin-security*: Adds options to limit upgrades to those for security.
- Many, if not all, plugins work automatically, so they do not include any man pages or help. However, you can see which plugins are installed on your system with the command *yum search yum*.

Very occasionally, you might find that a plugin, or perhaps a conflict between two or more plugins, prevents Yum from working properly. If that happens, you can use the option *--disableplugin = [plugin name]* to help you troubleshoot by disabling a specific plugin. If you are in serious difficulties, you can use the option *--noplugins* to run Yum without any plugins so that, with any luck, you can recover.

Other Package Managers

Yum is not the only RPM-based package manager available. Mandriva uses a similar tool called *urpmi*, and openSUSE uses *Smart*, a manager that not only resembles Yum but can install from Yum repositories. Other package management systems include Gentoo's *Portage* and Conary, which include version control that allows you to install different releases of the same packages.

However, Yum is almost certainly the most commonly used package manager after Debian and Ubuntu's *apt-get*. Furthermore, because Yum was developed years after *apt-get*, its developers had the chance to learn from *apt-get* and improve on it.

Although you probably won't see much in Yum that isn't in *apt-get* and its related utilities, you will find a less haphazard organization. In fact, if you know *apt-get* and are encountering Yum for the first time, you will likely be struck by how structured Yum is and how much easier it is to learn than *apt-get*. As a program, Yum is an intelligent innovation – and as a remedy for dependency hell, nothing short of a godsend. ■