YUM: Yellowdog Updater, Modified

Robert G. Brown
Duke University Physics Department
Durham, NC 27708-0305
rgb@phy.duke.edu

December 17, 2003

Contents

1 Introduction

Yum is a tool designed to provide a user or systems administrator with the ability to fully automate all aspects of Red Hat Package Manager (RPM) package installation and administration on RPM-based systems (both Linux and otherwise). It can be viewed as "competing" with package management tools such as Red Hat's up2date, Debian's Advanced Packaging Tool (apt-get), Mandrake's utility for managing and installing RPMs, (urpmi) and apt4rpm. If you like, its primary purpose is to prevent "dynamic link-library hell" on RPM-based systems while providing users and administrators a much higher level interface to RPM packages and repositories than that provided in the basic RPM .

Yum, however, is not YAAPT (yet another advanced packaging tool). Package management is far from trivial, and each of these tools has been driven by a distinct philosophy. Yum has from its post-yup beginnings been designed and improved strictly by *experienced systems administrators* managing *large networks of systems*. Its development is non-corporate and distribution agnostic, and as a *client pull tool* that works from *package repositories* it supports a *decentralized management model* that fits well in the Universities, government labs, and cluster environments whose administrators have primarily contributed to its development.

Yum is still a relatively young tool – it lacks a graphical user interface (GUI) for example, because its primary application in administered networks really doesn't need one – but is quite feature mature and is very portable. It is also under extremely active development, with features planned that will make it even faster and more efficient.

1.1 Article Organization

This section tells you how to avoid reading much of this article if you are in a hurry by describing its organization in more depth than that permitted by a table of contents. Readers of this article are likely to split into three general groups:

Users (of the Yum Client only): You have just installed a system that has yum preinstalled or want to yum-maintain a system already installed from some RPM based distribution. You may want to read the section on "What Yum Can Do" (a review of its principal features and advantages) and "Getting and Installing Yum" and then skip ahead to the section on "The Yum Client".

Systems Administrators: You are interested in setting up one or more RPM repositories and using yum to maintain a LAN of systems installed and maintained from these repositories (possibly augmented in various ways by public repositories that support yum). You are the primary audience for whom this article is intended, and will need to read all the sections.

Potential Yum Developers: You are in either of the two categories above but have or wish to develop python skills and are excited at the prospect of helping to create a that could one day make Linux "transparent" across all distributions and packaging schema. You too will need to read the entire article – for starters – and then of course go on and read all the documentation you can get your hands on, including the source.

Everybody can read or skip the Conclusion as they like – it contains a measure of future prognostication but little functional information essential to the use of yum in its current revision.

1.2 Useful Links

Yum Website: http://linux.duke.edu/projects/yum

One stop shop for yum and information about yum.

Yum HOWTO: http://www.phy.duke.edu/~rgb/General/yum_HOWTO.php

The yum HOWTO (probably still somewhat incomplete).

Yum article: http://www.phy.duke.edu/~rgb/General/yum_article.php

This article. (Self-referentiality! That's a neat trick...)

DULUG: http://www.dulug.duke.edu

Duke University Linux User's Group, host to several public mirrors and useful resources.

Linux@Duke: http://linux.duke.edu

Current host of the yum website, among other things.

The Linux Documentation Project: http://www.tldp.org

The name says it all. The place to go if you don't understand any tool referenced in this article, or to learn how to set up e.g. a web server or ftp server or network file server.

1.3 Yum History

Some years ago Dan Burcaw at Yellowdog Linux (an RPM-based distribution targeted at Apple Macintoshes) together with Bryan Stillwell, Stephen Edie, and Troy Bengegerdes created the Yellowdog Updater (yup). Yup extracted dependency information from RPM headers and constructed full dependency trees, permitting the automation of RPM installation and maintenance. On the strength of its features, Duke University (especially the Physics Department) came to rely heavily on yup.

However, yup was extremely slow. Operating on the client side, it had to completely download RPMs in order to extract critical header information to resolve all possible dependency loops. The header is only a tiny fraction of most

RPMs so this was obviously very inefficient. Powerful as it was, yup was also missing lots of obviously desirable features.

As a GPL project it was also easy to get involved and work to improve it. Seth Vidal, the Duke University Physics Department's highly talented systems administrator, took on this task. At first Seth worked to improve yup, but soon it became apparent that the tool would have to be completely rewritten in order to make it faster and more powerful, so at some point he split off from yup and created *yum*, the "yellowdog updater, modified". The resulting tool was an instant hit.

Seth was later joined by a physics post-doc, Michael Stenner, who took on the development of a generalized "urlgrabber" that manages most of yum's remote file access transactions, and many others (mostly systems administrators of large networks and beowulf cluster that now use yum as a primary maintenance tool – see the list of contributors on the yum project page for a complete listing).

With active input from many highly talented administrators and a very energetic open source development team, yum has become a very powerful tool indeed, capable of working such dark magic as safely upgrading a system from one distribution release to the next *while the system is operating*, as well as installing packages, updating packages, providing information about installed and available packages, and much more, all while operating many times faster than yup was able to do.

At this point, yum is supported and distributed via DULUG's public web resources and the Linux@Duke project (see links above).

1.4 Acknowledgements and Disclaimer

The author would like to thank the members of the yum mailing list, especially Seth Vidal and Michael Stenner for creating yum, Russ Herrold for taking a first stab and some of its documentation and Jonathan Pickard (co-author of the nascent Yum HOWTO). The good stuff in yum is due to their hard work, the errors in this article are my own.

The trademarks mentioned in this article belong to their owners, the opinions stated are those of the author, the "facts" presented in this article are open to question and may in fact be incorrect. What do I know, after all? Use at your own risk.

2 What Yum Can Do

"Yum" is really a set of python libraries with two primary command line user interface programs (yum and yum-arch) built to support installation and maintenance actions using RPMs provided in a *repository*. A repository can be any of the following:

- A website containing RPMs
- An FTP site containing RPMs

- A directory path (e.g. NFS mounted or local) containing RPMs.

(all of which can be thought of in generalized terms as Universal Resource Locator paths: URLs). A typical network-based repository might contain a mirror of a primary distribution and hence (in the case of Red Hat) directly support kickstart and network installation in addition to yum-based maintenance.

However, yum supports *multiple repositories.* Repositories can be added for many reasons:

- To extend the set of available RPMs. Two repositories can present the *union* of all the RPMs therein for installation and update purposes. It supports policy choices to use the order of the repositories in /etc/yum.conf to determine what to install or to always choose the latest revision.
- For robustness and redundancy. Yum supports a repository fallback scheme so if one repository intended to serve some purpose is unavailable, it automatically uses a fallback repository (perhaps a mirror of the first) again according to a controlled policy.
- As ordered *layers.* Yum can easily overlay, for example, a set of department-local RPMs on top of a larger set of organization-global RPMs). This permits proprietary packages to be made available only to selected clients for installation or maintenance purposes, or permits the overlay of RPMs containing private configuration information.
- Support *different purposes* in a maintenance and installation scheme. One department can be responsible for a repository that contains all the database rpms, another for all the graphical utilities, but both can be shared across the parent organization.

The ability to intelligently use multiple hierarchically ordered repositories to maintain client installations is one of yum's best features. Repository creation and maintenance is hence a major component of yum's value in a scalable administrative setting.

2.1 `yum-arch`: The Repository Tool

Yum currently consists of two separate components. One component of yum must be run on each repository the yum client seeks to use. Yum-arch is the tool that "yummifies" a repository, preparing it so that the yum client tool can easily and efficiently obtain the header information it requires to function. Yum-arch has a variety of debugging, informational, and security switches that are primarily of use to systems persons setting up repository and are well-documented in its man page.

2.2 `yum`: The Client Tool

The other component is yum itself, the client tool that provides all its direct functionality. Yum in its most current incarnation can:

- List packages (installed and available on all listed repositories) and their description and contents. As of recent versions, package groups too.
- Permit the easy installation of a package (or package group) and all of its dependencies with a single simple command.
- Permit the easy removal of a package (or package group) and all its unique dependencies (the ones that are not required to support other packages being retained) with a single simple command.
- Automatically update all packages on a system to the latest revision numbers available on the repository(s) to which yum is directed. Again, dependencies are resolved and additional packages added as needed. However, it is the responsibility of the maintainer of the repository set to ensure that the revision numbers are consistently maintained through the set and to deal with conflicts such as dependency loops and plain old bugs as they are revealed.
- Automatically upgrade an entire distribution from one distribution version to another (to the maximum extent that this is possible – it isn't always possible or wise to proceed this way).

Of these, probably the single function of the greatest importance is also the most mundane – automated updating. While the other functions are all very useful and convenient, these days it is *essential* to keep all clients in a network fully patched and current in order to keep them secure. However, this presents a serious dilemma – few organizations are structured so that any single administrative group has root privileges on *all* the computers in the organization, and in most cases this is a *good thing*.

Yum, however, is a network *client pull* tool and requires *no centralized privileges* on the client in order to function. Each client contacts the repository server and requests the download of all files the client requires to update itself. These downloads can be anonymous and public or authenticated by any mechanism supported by yum's urlgrabber utility. By scripting a nightly yum update, all of the students, the faculty, the staff, and the various departments of a University can maintain an interval of less than 24 hours between placing a critical security updated RPM in the repository and having it propagated to all the client systems, all without having any individual with "super-root" privileges on all of the systems being updated.

3 Getting and Installing Yum

3.1 Root Privileges, Editors and Terminal Commands

If you are an experienced administrator, you can skip the following minilecture on root, terminals, and editors except to note that to install yum and for many, if not most, of yum's commands to work properly it is necessary to become root (the superuser). Privileged commands will generally be indicated by means of the '#' prompt in the examples.

If you are a linux user, you may have used root privileges only via a Graphical User Interface (GUI) tool. At this moment there is no suitable GUI for yum, so yum commands must be entered at a *command line* (terminal).

Ways of bringing up a terminal window vary a bit from system to system. On some a terminal might be available as a Gnome or KDE menu entry. On others it might be on the bottom task bar, or as a menu selection if you right-click and hold with the focus over the X background. Terminals might be named "terminal", or xterm, or rterm, or kterm. Nearly any of these will suffice, although I personally prefer plain old xterms.

To begin to work with many of the examples, you will need to start up a terminal and become root on the terminal. To become root, enter:

```
$ su
```

and enter the root password when prompted to do so. The prompt should change from the usual default $ to the sharp sign '#'. Tradition holds that this is to remind you to be sharp as root.

Note Well: As the root superuser you can *utterly destroy* your system's linux installation with a single careless keystroke! Be *very certain* that you type things in correctly, especially if they involve wildcard characters such as * or ?.

You will also need to use some sort of editor to create and modify some of the configuration files mentioned in this article. Your GUI setup will likely have at least some editors, or you can use an old Unix standby such as emacs or vi from the command line. These tools are not terribly easy learn and instruction is well beyond the scope of this article. Seek help from man pages, online resources, or The Linux Documentation Project (linked elsewhere in this document).

3.2 Preparing Your System for Yum

Yum is a designed to help you maintain a system relative to a well-maintained rpm repository. In order for this to work, it is important to clean up your system so that its RPM database is consistent with what is actually installed on your system. Installing yum on a brand new, clean system installed from RPMs is ideal. If your system has been in use for a while, though, don't panic. Yum may install and work just fine.

However, if you have ever used the `--force` and/or `--nodeps` options to install RPM packages, or if you've built and installed a number of programs (perhaps into /usr/local) from source, you may encounter problems. In these cases the best thing to do is to do your best to remove the offending packages or programs and reinstall them from a consistent set of RPMs built for the base distribution you are using.

3.3 Installing Yum

Given this, it should come as no surprise that in order to install yum, you should visit its primary website:

`http://linux.duke.edu/projects/yum`

and download RPMs for the latest stable version that match your installed version of rpm and python. Some of the features described in this document (support of package groups, for example) are available only in version 2. Individuals interested in participating in the development process or seeking additional support are encouraged to join the yum list on this site.

Once the appropriate rpm is downloaded, it can be installed by using a command like:

```
# rpm -Uvh yum-2.0.4-1.noarch.rpm
```

If rpm complains about missing dependencies, try to install or upgrade the missing components or fall back to a version of yum that better matches the components you already have installed or available.

If yum is initially installed from rpm, then yum can thereafter update *itself* as well as everything else, so this is by far the preferable way to proceed. However, on the primary yum site there is a tarball (`.tar.gz` source file) and links to the public CVS repository for the project. Individuals who want to look over the sources, create their own RPM, or participate in yum development can of course download the sources and rebuild yum locally.

4 Creating a Simple Yum Repository

It is very simple indeed to set up a yum repository. There are only three steps. We will briefly outline them below, making the assumption that you are going to set up a mirror of the Fedora core repository on the DULUG (Duke University Linux User's Group) public Red Hat Mirror:

`http://mirror.dulug.duke.edu/pub/fedora/linux/core/1/i386/os/`

Note that this path (and the public accessibility of this server) are both subject to change. Modify the path and scripts below to mirror the particular repository of your choice, or create your own repository from (for example) CD images for your favorite RPM-based Linux distribution. **Note Well** however, that if you create a "static" repository directly from CD/ISO images, yum will not be able to perform one of its very important functions – managing package updates to keep your system current and secure. In this case you might consider using your static repository for a initial installs and "package shopping", but put additional repositories to provide you with updates into the yum.conf on your clients (see the section below on "layering yum").

Note also that if you are not interested in setting up your own repository, yum will still function as a client on your workstation(s) and server(s) if you can find a public repository that supports yum. Many of them do, in particular the Fedora repositories. If this describes you, you can skip all of this section except the part on getting yum.

A reasonably complete list of public repositories is available on the yum project site. Systems administrators caring for large numbers of systems are strongly encouraged to set up their own local mirrors of these repositories to (and if possible to make them publicly accessible) to minimize the load on the mirrors.

4.1 Mirroring a Repository

Let us as a demonstration create a mirror of a Fedora repository that can be used to install and maintain a Fedora-based LAN (both workstations and servers). For the sake of being concrete, it will be assumed that the repository is being created on a web server, although it might equally well be created on an FTP server or NFS server (or the same repository exported all three different ways to different clients). This article will *not* detail the process of setting up any sort of server initially; support for this is readily available from many books and in HOWTOs provided by *The Linux Documentation Project*:

`http://www.tldp.org`

The following is a trivial script that uses *anonymous rsync* to make a copy of the entire fedora core repository. Note that you will have to install rsync for this to work! Rsync is available as a standard rpm in most distributions and on many public repositories and mirrors. If necessary, obtain and install rsync before attempting to use it to create a mirror.

When this is done, enter this into a file named (for example) `rsync.fedora` and use `chmod 755 rsync.fedora` to make it executable.

```
#!/bin/sh
rsync -av --partial --stats \
  mirror.dulug.duke.edu::archive/pub/fedora/linux/core .
```

Place this script on an executable path or prepare to execute it by explicit path.

Note that one can add an `--exclude` argument to this command if one wishes to omit part of the tree for any reason. For example, adding `--exclude isolinux` after the `--stats` argument causes the isolinux directory under the fedora core to be omitted. If you don't plan to turn the mirror into a bootable cd, this directory is of no particular use to yum, although it is also not terribly large. This exemplifies how the distribution can be further trimmed or expanded according to your needs by editing the script. As long as the RPMs themselves are mirrored, yum itself will function.

On the server, create a suitable directory path and rsync mirror the repository using the script. For a web-based repository, this might be done with the following commands:

```
$ cd /var/www
$ mkdir fedora
```

```
$ cd fedora
$ rsync.fedora
```

This script will typically take a fairly long time to complete the first time, depending on your bandwidth and the load on the Fedora mirror server. Afterward, however, running the script will only cause *changes* in the repository to be downloaded.

In many cases one will want to create a short nightly or weekly cron script to automate a regular resynchronization of your mirror with the original. You may also want to add a step to check the MD5 or GPG signatures of the packages (if any are available) to ensure that they have not be tampered with.

4.2 Yummify the Repository

In many cases you will no longer have to yummify a repository you mirror (where by "yummify" I mean prepare the repository so that yum clients can work their magic when updating or installing software from them). Yum is being widely adopted by the major RPM distributions so that a mirror of the Fedora repository given in this article will very likely already be yummified and will equally likely automatically install yum as a part of the base distribution. However, you need to know *how* to yummify a distribution, as you may want to set up your own repository of RPMs you've built or collected yourself, or you may need to re-yummify a repository after updating or adding RPMs to it.

This is, fortunately, extremely simple. Assuming that you've installed your repository on the primary path `/var/www/fedora` as in the running example, simply run:

```
$ yum-arch /var/www/fedora
```

That's it. Yum-arch will descend into the fedora tree, looking for RPMs, extract their headers (with all the requisite dependency and path information) and structure it so that it can be delivered to any yum client without the need of downloading the RPMs themselves. It is possible to have more than one RPM directory in the repository; all RPMS in the tree will be yum-accessible. However, it is not possible to mix architectures, and is a generally bad idea to even mix RPMs built for different revisions.

Note that there are a fair number of options that yum-arch recognizes, most of which aren't terribly important for beginners to be aware of. The one possible exception is -c, which causes yum-arch to check GPG and/or MD5 checksums, if it can. Otherwise you should (as always) read the man page and other documentation before experimenting with options.

4.3 Export the Repository to the Client(s)

As noted above, yum can use repositories provided to the clients via the web, via ftp, or via any filesystem path (including those provided by a variety of network base filesystems). There are therefore many choices for servers, and it is far beyond the scope of this article to describe how to set each possibility up. We will therefore *assume* that you are running a straightforward apache webserver and need to export the Fedora path above so that it becomes `http://my.webserver.org/fedora` on your web server.

To do this, add a section such as:

```
<Directory "/var/www/fedora">
    Options Indexes Includes FollowSymLinks
    AllowOverride ALL
    Order allow,deny
    Allow from all
</Directory>
```

to /etc/httpd/conf/httpd.conf in an appropriate place and restart the webserver:

```
# /etc/init.d/httpd restart
```

Then, as a preview of sorts, create a yum.conf entry that points to this repository:

```
[fedora-core]
name = Fedora Core
baseurl=http://www.myserver.org/fedora/
gpgcheck=1
```

and set it aside for the next section.

Note that the `gpgcheck=1` line causes yum to check signatures of all the files it looks at, which is a good thing to do, at least if you wish to minimize the risk of installing a trojanned RPM that might have been inserted into one of the repositories you manage or mirror on every system in your enterprise.

Note also that there can be **no leading whitespace** in /etc/yum.conf – leading whitespace is interpreted by the configuration file parser as a line continuation (which is in fact used to add more than one baseurl for fallback purposes).

Warning! If the repository is a locally mounted filesystem, remember to include the leading path slash for the URL, as in

```
baseurl=file:///path/to/repository
```

Forgetting to do this is a fairly common cause of initial failure.

That's all there is to it!

In some ways yum is a tool designed for systems administrators (even if they care for only one system) which is why this article started by describing the setup of a yummified repository. Once you *have* a repository, however, it is time to learn to use yum itself as an installation/maintenance client for RPM repositories. This is the topic of the next section.

5 The Yum Client

If you are a systems administrator or Linux geek you've likely set up a repository of one sort or another for testing by now. Others, who have installed a pre-yummified distribution from one source or another and only want to know how to set up their yum clients, will have skipped the previous section altogether. Welcome back!

For starters, let's make sure that *everybody* has access to a yum repository, whether or not they built one. Chances are decent that if you installed yum from the main yum site:

`http://linux.duke.edu/projects/yum`

yum will "work" in that it will function at the client level if the client has access to the Internet and hence the default repositories in its /etc/yum.conf file. We will therefore begin by testing it.

As root, enter:

```
# yum list
```

at the # command prompt in a tty window (for example, in an xterm). With luck, you will see something like:

```
rgb@lilith|T:7#yum -c /tmp/yum.conf list
Gathering header information file(s) from server(s)
Server: Yum Test
Finding updated packages
Downloading needed headers
Name                                Arch   Version                 Repo
--------------------------------------------------------------------------------
xmlsysd                             i386   0.2.0-1                 yum-test
...
```

where there can be a *lot* of packages in the list (I'm using a tiny two RPM repository with its own custom yum.conf in this example, as you can see, to avoid this). These are all packages that are on the repository but not yet installed on your system!

Don't be discouraged if instead of this you get some sort of error message, or even if error messages persist through several more of the steps outlined below. Yum is a *very conservative* program, and it tries very hard to never do anything that will break your system with a default action. Some of its warning messages (especially ones you will get if your RPM database is inconsistent or corrupted by the use of RPMs `--force` or `--nodeps` options to install packages in the past) should be fully resolved by "cleaning up the system" before yum will run without complaint. As noted above, yum is happiest when run on a clean install from a yummified distribution repository, and your system will remain happiest if *only* yum is used to add or remove *only* packages from a set of yummified repositories that are consistent with this initial distribution repository, except when yum itself is used to upgrade to a different one.

But how can we arrange for this to be so? And once we've made the arrangements, how can we use yum for routine installation and maintenance? The remainder of this section contains two subsections: one on yum.conf, yum's configuration file that tells it, among other things, where to look for RPM repositories, what order to use them in, and what kind of policy and security features you wish to use when installing or maintaining packages from them; and one on the yum client itself.

5.1 yum.conf

There are three ways to get a working /etc/yum.conf file. The best way is to get one "automatically" (or at worst, to have to pick one that matches your installation from a list of pre-built alternatives). If you're using yum from an RPM installed from a repository on your site (for example) it very likely is pre-configured with a /etc/yum.conf file that directs it to just the right campus or corporate or lab repositories.

Alternatively, there is a list of pre-built yum.conf entries for yummified public repositories on the primary yum site:

`http://linux.duke.edu/projects/yum/repos/`

Chances are pretty good that you can just grab one that matches your distribution (or that you'd like to upgrade to from your existing distribution) and install it. To use them, click on them and paste them into your yum.conf files as described next.

The last way is to build your yum.conf to meet your specific needs and support your own local repositories. This is the route that will almost certainly be taken by most systems administrators who build their own repository. To do this, start by *reading the yum.conf man pages* that should have come with your distribution. *Look over* the default /etc/yum.conf that was installed by the RPM. Finally, browse through the additional examples below. Chances are pretty good that one of these will resources will provide a template that is adequate to get you started, and once you get things running at all a bit of experimentation will soon make you an expert.

So let's look at a yum.conf file that is fairly minimal – just enough to get you started:

```
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log

[fedora-core]
name = Fedora Core
baseurl=http://www.myserver.org/fedora/
gpgcheck=1
```

Systems persons who are building a fedora mirror as we proceed should note that this yum.conf uses the repository we "built" in the previous section, where of course the actual URL should point to that repository.

Users who want to accomplish the same thing could instead paste in an entry from the list of public yum repositories linked above. For example, here is a yum.conf file containing several repositories from that list:

```
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log

[fedora-stable-9]
name=Fedora Project Stable RPMS for RHL 9
baseurl=http://download.fedora.us/fedora/redhat/9/i386/yum/stable/
gpgcheck=1

[fedora-updates-9]
name=Fedora Project update RPMS for RHL 9
baseurl=http://download.fedora.us/fedora/redhat/9/i386/yum/updates/
gpgcheck=1
```

From the comments, one could hope that these two repositories contain what is effectively Red Hat 9, as ported into the Fedora project, plus updates. Note that we've turned on gpgcheck under the assumption that this is a good thing to do to ensure that the RPMs we grab haven't been trojanned.

Note that yum has *many* options that can be selected in the yum.conf file, only a *few* of which are illustrated in these examples. Each baseurl can have additional fallback repositories for additional stability and robustness. gpgchecking can be enabled (causing each rpm to be examined to ensure that it is correctly signed). The debuglevel can be cranked up or down to help diagnose problems. pkgpolicy can be set to determine the order in which yum chooses to decide between two versions of the same package on different repositories it is using at the same time. Packages can be listed to *exclude* them from the automated update process. The failover method yum uses to select a server to use from a list of servers in a baseurl can be chosen. More options may have been added since this article was written, or old options may have gone away.

So *read the man pages* on yum.conf. It is the *only* thing likely to be authoritative.

At this point we will assume that you are on a client, that you have a functional /etc/yum.conf installed that directs your system to a repository (that you built yourself or not) that is consistent with your system's existing RPM distribution, and that `yum list` *works* (as root) to list packages on the repository. Now let's see what yum can do.

5.2 yum

At last we should have a system on which yum is functional. For systems administrators building their own repositories and hand crafting yum.conf's to access them, it may have been an arduous and long journey (well, not THAT long – one can actually create a directory path, throw some RPMs into it, run yum-arch on it, cut-n-paste a basic yum.conf into existence, and run yum -c yum.conf list on your new "repository" in about five minutes). For system owners who got a pre-configured yum automatically installed on their system ready to run, well, you may have just skipped to this point (but at least glance over all the sections above so you can appreciate all the work the systems administrator who put together your repository base and the yum rpm did for you).

Note Well: If your system's RPM database is inconsistent because of the use of RPMs `--force` or `--nodeps` options, or because you've built and installed libraries or programs directly from source (as opposed to source RPMs) then yum may complain, not work correctly, or even "break" some installed subsystem as it attempts to update or install a package that you've already installed and configured by hand. Yum works best on a system that has been installed and modified *only* with RPMs, ideally RPMs built and distributed in a consistent set. As you can see, this bears repeating...

We now proceed to review the basic yum commands and what each one does for you. This section is *not exhaustive* by any means. Yum is a very powerful tool with a variety of primary actions that can be modified by both command line options and configuration settings in /etc/yum.conf, so to learn its full range of power you will need to *read its man page* and possibly take a tour of the yum HOWTO. The following should suffice, though, to get you started and may be all you ever need.

All yum commands cause yum to take the following actions:

- The *first time* yum is called, **as root**, it downloads all the header files in all repositories in its yum.conf and puts them in a cache on your system. Subsequently it only downloads ones that have changed since the last time the cache was refreshed.
- It then reads your RPM database (which should be consistent, see note above) to determine what packages are installed.
- Determine which yum command has been invoked, and process the command.
- If the command is run interactively (the default) yum will prompt the user for permission to proceed before actually updating or installing files.
- If yum is run with the `-y` option, yum will assume a "yes" answer to all interactive questions. This is typically used in a script or to process "known safe" installs or updates.
- If yum is run with the `-d 0` option, it will run as quietly as possible. `-y -d 0` are typically invoked together in a scripted update where only serious errors are to be reported.
- There are many other yum action modifiers; read the man page for a synopsis.

Yum is *very conservative*. It will *never* perform the equivalent of a `--force` install. It was designed, as noted above, by systems administrators to facilitate the *consistent and scalable* maintenance of systems, and thus will balk at any action that might cause your system to enter "package hell" (a state of package inconsistency). If yum refuses to do something, then the thing it refuses to do is probably a bad idea for most people to think about. If you are skilled enough to know that it is really OK on YOUR system, you are skilled enough to use rpm itself to override yum.

But do you really want to? Perhaps what you really should be doing is rebuilding the RPMs in question, or fixing some apparent conflict instead of ignoring it. In a sense, yum is the ultimate sanity check. If yum is happy, your repository is consistent, your RPM database is consistent, and your system is happy. If yum is *not* happy, then chances are pretty good that at least some part of your system *could* break, which would cost you a lot more time to set right than just making everything consistent to start with.

The following is a brief synopsis of the commands:

`yum install package1 [package2 package3...]`: Yum checks to see if package1 is already installed and is the latest version. If not, it downloads package1 *and all its dependency packages* (saving them in its cache directory) and installs them.

`yum update [package1 package2...`: If there is a basic, essential yum command this is it. Without arguments, yum compares *each* installed package to packages available on the repository list. If a newer package is available in the repository, it downloads it (saving it in the cache directory) and updates the old one. If a list of packages is given, it only checks for updates of those packages (and their dependencies).

Note that RPM updates generally preserve configuration information and restart daemons and so forth, so your system will generally not require a reboot, although updating the kernel is an obvious exception. Kernel updates are a special case and systems administrators will generally set a policy on this in /etc/yum.conf. In any event, yum will not reboot your system even after a kernel update.

`yum remove package1 [package2 package3...]`: Yum will first check to see if removing the package breaks anything else (that is, if anything else already installed depends on this package). It will then offer to remove the package *and all the packages that depend on it.* Yum will generally refuse to leave "dangling" packages that won't work anyway because a critical library or component has been removed. Note that running remove with the `-y` option is probably a really stupid idea for most people, unless they are *certain* that they know all of the possible side effects.

`yum list [ ,available,installed,updates,extras] [package list]`: This is one of yum's most useful commands. Run without arguments or a list

of packages, yum lists all packages available to be installed on all repositories. Run with a list of packages (which can contain file wildcard globs, e.g. `yum list kernel`
to list all kernel package) it shows *both* the installed packages *and* the available packages that match the expanded list! The `installed`, `available`, and `updates` options show only installed packages, available packages, or packages that will be updated by the next `yum update` command. **Note Well** that when run interactively from a shell, wildcard characters need to be escaped from the shell itself or the command will have unexpected consequences.

`yum info [ ,available,installed,updates,extras] [package list]`: This command extracts the description and summary from all matching packages and prints it out. It takes the same options as list, but provides different information. `yum info  > /tmp/packages` thus provides you with a *shopping list* in /tmp/packages of *every package* either already installed on your system or available on its set of repositories! This can be very useful indeed!

The `extras` option lists all packages that are installed but *not* available in the repositories. These are packages you may have installed by hand, for example. A good practice is to create a repository for these local RPMs (which can be nothing but a local directory on the system) and add the repository to the yum.conf file for the system, following the recipe above.

`yum search keyword`: Searches for packages matching the keyword string in the description, summary, packager and package name fields. This is useful for finding a package you do not know by name but know by some word related to it. The keyword string can contain wildcard file globs. Be sure to escape these from the shell as noted above if run interactively.

`yum provides [feature|file]`: Find package(s) that provide some feature or file. Accepts file wildcard globs, which again should be escaped from the shell if run interactively.

`yum clean`: Cleans up yum's cache file. On systems with small disks or limited /var resources, the RPMs that yum caches can fill up the disk. Running `yum clean` periodically can free up the resources, but it can also mean that yum runs more slowly when doing certain things. Clean takes various options: see the man page for details.

`yum group[list,install,update]`: Yum has recently acquired the ability to perform the list, install, and update functions on *package groups* defined for a system, with otherwise similar arguments. This makes it relatively easy to install an entire block of software all at once, on systems that have package groups defined.

There are a few more commands (some of them deprecated) and of course many more options. Also, if you installed an relatively early version of yum (perhaps in order to support an older Linux distribution that you don't wish to upgrade with the latest python and rpm required to support the latest version of yum) some options may be missing! Read the man page of the particular yum version you installed to review all available commands and options.

5.3 Automating Nightly Updates

Note that the commands above fall into three broad categories: *informational* command, *installation* commands, and *update* commands. The first two are almost always run interactively. `yum update` needs to be run *regularly* and hence needs to be automated. The yum rpm recognizes this by installing /etc/cron.daily/yum.cron and /etc/init.d/yum. The former is a script that runs yum update. The latter is a control structure that permits this action to be controlled via `chkconfig`.

Yum may have been installed on your system with the nightly cron already enabled by (for example) a departmental or institutional administrator who repackaged yum with a local /etc/yum.conf that points to a local set of repositories. At Duke, for example, any Linux system installed from the campus repository already contains yum as part of its Basic package. Yum itself is preconfigured to *both* update and install future RPMs from the same repository as the original system install *and* to update the system to the latest RPMs placed on that repository every night.

This makes the campus security officer very happy, as an exploit can be discovered in a Linux package in the morning, a patched RPM that closes the exploit can be dropped into the repository at five in the afternoon, and *every Linux system on campus* will be updated to the patched package and secure again by the following morning. It also makes the non-expert users of these systems very happy, as if they report a bug or problem that can be easily patched and the relevant RPM updated and dropped back into the repository, they can do "nothing" and the problem is resolved the next day, even though the central repository administrator may have no direct access to their systems at all.

If you are using yum in a similar context inside an organization that maintains a repository and pre-configured yum to use this repository according to some policy, it is likely wise to leave the setting alone (or at least find out what the policy is before changing it). If you are a systems administrator seeking to *create* such a setup for your organization, hopefully this document is sufficient for you to see how to go about it (further help is available on the yum list if this is not correct).

If you are an ordinary user seeking to set up automated maintenance from a public repository, then *after* you have set things up according to the general instructions above so that a `yum update` reports that all packages are installed and the latest version, you will likely want to turn on the nightly cron update. Simply become root and enter:

```
# chkconfig yum on
```

```
# /etc/init.d/yum start
```

(where the latter is necessary the first time, as chkconfig does not start services as they are turned on). On all subsequent reboots the yum nightly cron will be automatically enabled at boot time.

It is worthwhile to look over the nightly cron script in /etc/cron.daily/yum.cron. As of this moment, it contains:

```
#!/bin/sh

if [ -f /var/lock/subsys/yum ]; then
        /usr/bin/yum -R 10 -e 0 -d 1 -y update yum
        /usr/bin/yum -R 120 -e 0 -d 1 -y update
fi
```

Note that this tells yum to *first* update yum, *then* update everything else (a wise order to follow, as one doesn't want yum's guts to be replaced in midstream). The `-R window` option causes yum to wait a random amount of seconds in the window and then run – this causes the server load associated with e.g. an entire campus of updates running out of cron at the same time to be spread out in time a bit. Systems administrators or users may wish to tweak these settings a bit for their local environment or according to the policy dictates of the public repositories they are using.

5.4 Examples

One of the best ways to learn any tool is by example. The following are some actual verbatim examples of simple yum usage. Imagine that you want to add some simple speech synthesis capabilities to your PC so that it can issue verbal prompts or read simple messages out loud. Looking through the info on your repository collection, you discover the *festival* package, which does just what you need (note that we are root):

```
rgb@lucifer|T:11#yum info festival
Gathering header information file(s) from server(s)
Server: Linux@Home 9 - i386 - Base
Server: Linux@Home Distributable add-on packages - 9 i386
Server: Linux@Home Nondistributable add-on packages - 9 i386
Server: Linux@Home personal add-on packages - 9 i386
Server: Linux@Home NFS test - 9 i386
Finding updated packages
Downloading needed headers
Looking in Available Packages:
Name   : festival
Arch   : i386
Version: 1.4.2
Release: 16
```

```
Size   : 58.35 MB
Group  : Applications/Multimedia
Repo   : Linux@Home 9 - i386 - Base
Summary: A speech synthesis system.
Description:
 Festival is a general purpose, multi-lingual speech synthesis system
developed at CSTR. It offers a full text to speech system with various
APIs, as well as an environment for development and research of speech
synthesis techniques.

It is written in C++ with a Scheme-based command interpreter for
general control.


Looking in Installed Packages:
```

Hmm, available, not installed. Let's install it.

```
rgb@lucifer|T:12#yum install festival
Gathering header information file(s) from server(s)
Server: Linux@Home 9 - i386 - Base
Server: Linux@Home Distributable add-on packages - 9 i386
Server: Linux@Home Nondistributable add-on packages - 9 i386
Server: Linux@Home personal add-on packages - 9 i386
Server: Linux@Home NFS test - 9 i386
Finding updated packages
Downloading needed headers
Resolving dependencies
Dependencies resolved
I will do the following:
[install: festival 1.4.2-16.i386]
Is this ok [y/N]: y
Getting festival-1.4.2-16.i386.rpm
Calculating available disk space - this could take a bit
festival 100 % done 1/1
Installed:  festival 1.4.2-16.i386
Transaction(s) Complete
```

Nothing to it. We wonder if there are any other festival components and if we need them:

```
rgb@lucifer|T:13#yum list festival\*
Gathering header information file(s) from server(s)
Server: Linux@Home 9 - i386 - Base
Server: Linux@Home Distributable add-on packages - 9 i386
Server: Linux@Home Nondistributable add-on packages - 9 i386
```

```
Server: Linux@Home personal add-on packages - 9 i386
Server: Linux@Home NFS test - 9 i386
Finding updated packages
Downloading needed headers
Looking in Available Packages:
Name Arch Version Repo
--------------------------------------------------------------------------------
festival-devel i386 1.4.2-16 duke-base

Looking in Installed Packages:
Name Arch Version Repo
--------------------------------------------------------------------------------
festival i386 1.4.2-16 db
```

Looks like there is a development package. This could be a library so that one can build one's own speaking applications, or it could be examples and data files. It isn't, apparently, a dependency of festival (or yum would have automatically selected and installed it too). One wonders what is in it?

```
rgb@lucifer|T:16#yum info festival-devel
Gathering header information file(s) from server(s)
Server: Linux@Home 9 - i386 - Base
Server: Linux@Home Distributable add-on packages - 9 i386
Server: Linux@Home Nondistributable add-on packages - 9 i386
Server: Linux@Home personal add-on packages - 9 i386
Server: Linux@Home NFS test - 9 i386
Finding updated packages
Downloading needed headers
Looking in Available Packages:
Name   : festival-devel
Arch   : i386
Version: 1.4.2
Release: 16
Size   : 11.57 MB
Group  : Development/Libraries
Repo   : Linux@Home 9 - i386 - Base
Summary: Development files for the festival speech synthesizer.
Description:
 Development files for the festival speech synthesizer. Install
festival-devel if you want to use the festival speech synthesizer from
within your own progrmas and/or if you intend to compile other programs
using it.


Looking in Installed Packages:
```

Hmmm, looks like we might need this as well some day. Disk is cheap, maybe we should install it now. On the other hand, if we ever *do* need it, with yum we can install it in a matter of seconds. With yum, one can actually do things like install a tool you need only rarely, use it for five minutes, and remove it at the end to recover the space and other resources.

Let's finish off with an always useful:

```
rgb@lucifer|T:18#yum update
Gathering header information file(s) from server(s)
Server: Linux@Home 9 - i386 - Base
Server: Linux@Home Distributable add-on packages - 9 i386
Server: Linux@Home Nondistributable add-on packages - 9 i386
Server: Linux@Home personal add-on packages - 9 i386
Server: Linux@Home NFS test - 9 i386
Finding updated packages
Downloading needed headers
No Packages Available for Update
No actions to take
```

It's always a pleasure to know that every single software package on the system is current as of the last round of patches and security fixes mirrored to my home repository from Duke.

Now to go play with festival. Maybe I can tell the cat to stop sleeping on the keyboard from work...

6 The Future

As noted above, yum is in an extremely active state of development. Volunteers are writing a GUI for yum to make it in some sense auto-documenting and simpler for novices to use. The primary developers periodically ask the yum list for ideas for new features (although since most of those list members are systems administrators *using* yum to manage entire organizational-scale networks, they have not in the past been shy about requesting new features without prodding and often accompany the requests with patches to facilitate their addition). Bleeding edge participants can access the latest development snapshot of yum, which is updated as often as several times a day. With this sort of energy, it should come as no surprise that yum has an ambitious future.

Things on the yum TO DO list include:

- Improved extraction/transmission of repository metadata (e.g. package listings and dependencies) to make yum more efficient. Currently yum sends the header files for each package separately, and this is relatively slow and inefficient (order of a few seconds per yum invocation after the first). This should be reducible to less than a second for any high bandwidth connection, a bit longer for low bandwidth cable, DSL, or wireless connections.

- This process has been combined with an effort to *unify* the header metadata description across *all Linux packages* in an extensible way on top of XML. This is an extremely exciting development that could in the future flatten the currently significant barriers between installing packages built for different distributions.

- Efforts are underway to make yum an integral part of a system's primary installation process. There are reasons that yum would make a much better tool to bring a system into agreement with a specific list of packages and package groups than kickstart, for example. This too is discussed below.

- Efforts are underway to make yum retrieve and act on *source* RPMs as well as architecture-specific *binary* RPMs. This has extraordinary implications for the future, discussed below.

Kickstart (and Red Hat's interactive installation tool) are in many ways lovely tools but they have one major flaw. If they are interrupted at any time after the system configuration has been selected and the package installation process is begun, they have to be rerun from the beginning. Since they *can* take quite a long time to complete (installing whole *giga*bytes worth of programs and data over channels that might be as slow as a DSL link supporting tens of *kilo*bytes per second) there is a significant window where the system is vulnerable to interruption. Also, as a pure matter of convenience, one might wish to be able to "use" (or otherwise work on) a system in its basic configuration while its customizing packages are being uploaded and installed, especially if the latter will take hours to complete.

Yum will eventually (soon) permit these two steps to be completely separated. Indeed, this could be done now, but not easily as a "prepackaged" process. One will be able to kickstart (or interactively install or an install based on a PXE ramdisk)) a very basic system – one with "no" packages but the basic/default package – and then use yum to "finish" the install to bring it up to a predefined standard. If it is interrupted in the middle, it can just be restarted – yum will not duplicate or waste any of the work it has already accomplished but will just chug along until it is finished. In the meantime, the system can be worked on or with as it will be bootable and usable after the initial very short pre-yum installation process.

Making yum work at the source level is very exciting indeed. If yum were able to retrieve RPMs *and their dependencies* as *source RPMs* and automatically begin building and installing binary RPMs starting at the bottom of the dependency trees, it would be possible to *do away with binary rpm support* to whatever extent one desired. A system install could begin with a basic system plus the associated compiler, and then (with the help of yum) *build itself.* This would make it more or less *impossible* to build systems with problems associated with incompatible binaries or libraries, as those problems would be revealed and resolved at the *source* level and corrected in the primary repositories almost instantly.

This would not, in fact, consume an inordinate fraction of a system's resources to do. The *smallest* disks currently being sold are order of 40 GB in size, more commonly 80 GB or more. Caching a local copy of source and binary RPMs on a system occupies a modest fraction of that that will only diminish in time, and with yum one would only actually retrieve what one actually wanted to install.

This represents a genuine paradigm shift, and might well completely alter the way Linux and GPL/open source software in general are distributed and supported. A Linux distribution would become little more than a collection of source RPMs, with tools that would convert them into a binary RPM repository for convenience sake in a matter of a few hours. Provided only that metadata descriptions are sufficiently robust to cope with the network of revision dependencies that can exist between source RPMs, in short order the *same* basic set of source RPMs would represent stable Linux across all distributions.

To conclude, yum has already begun to alter the installation and update paradigm used to scalably maintain RPM-based Linux computers, but it in a sense has only begun to do what it is capable of doing. In the near future, yum and its contributed extensions may well alter Linux itself, and by extension the Internet itself, by providing a simple way of network-distributing *arbitrary software* in source/package form. Even compact disks are now proving clumsy as ways of moving around the many gigabytes of tools and data associated with a fully functioning Linux system, and at best they represent a base from which to work that necessarily must be kept up to date and patched in any event. The network will of necessity be *the* mechanism for this process of updating in the future, and yum is the best tool to emerge to date for supporting the entire process.

A License Terms for "YUM: Yellowdog Updater, Modified"

A.1 General Terms

License is granted to copy or use this document according to the Open Public License (OPL, enclosed below), which is a Public License, developed by the GNU Foundation, which applies to "open source" generic documents.

In addition there are three modifications to the OPL:

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder. (This is to prevent errors from being introduced which would reflect badly on the author's professional abilities.)

Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder. (This is so that the author can make at least some money if this work is republished as a textbook or set of notes and sold commercially for – somebody's – profit. The author doesn't care about copies photocopied or locally printed and distributed free or at cost to students to support a course, except as far as the next clause is concerned.)

The "Beverage" modification listed below applies to all non-Duke usage of these notes in any form (online or in a paper publication). Note that this modification is probably not legally defensible and can be followed really pretty much according to the honor rule.

As to my personal preferences in beverages, red wine is great, beer is delightful, and Coca Cola or coffee or tea or even milk acceptable to those who for religious or personal reasons wish to avoid stressing my liver. Students at Duke, whether in my class or not, of course, are automatically exempt from the beverage modification. It can be presumed that the fraction of their tuition that goes to pay my salary counts for any number of beverages.

A.2 The "Beverage" Modification to the OPL

Any user of this OPL-copyrighted material shall, upon meeting the primary author(s) of this OPL-copyrighted material for the first time under the appropriate circumstances, offer to buy him or her or them a beverage. This beverage may or may not be alcoholic, depending on the personal ethical and moral views of the offerer(s) and receiver(s). The beverage cost need not exceed one U.S. dollar (although it certainly may at the whim of the offerer:-) and may be accepted or declined with no further obligation on the part of the offerer. It is not necessary to repeat the offer after the first meeting, but it can't hurt...

A.3 OPEN PUBLICATION LICENSE Draft v0.4, 8 June 1999

I. REQUIREMENTS ON BOTH UNMODIFIED AND MODIFIED VERSIONS

The Open Publication works may be reproduced and distributed in whole or in part, in any medium physical or electronic, provided that the terms of this license are adhered to, and that this license or an incorporation of it by reference (with any options elected by the author(s) and/or publisher) is displayed in the reproduction.

Proper form for an incorporation by reference is as follows:

Copyright (c) <year> by <author's name or designee>. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, vX.Y or later (the latest version is presently available at http://www.opencontent.org/openpub/).

The reference must be immediately followed with any options elected by the author(s) and/or publisher of the document (see section VI).

Commercial redistribution of Open Publication-licensed material is permitted.

Any publication in standard (paper) book form shall require the citation of the original publisher and author. The publisher and author's names shall appear on all outer surfaces of the book. On all outer surfaces of the book the original publisher's name shall be as large as the title of the work and cited as possessive with respect to the title.

II. COPYRIGHT

The copyright to each Open Publication is owned by its author(s) or designee.

III. SCOPE OF LICENSE

The following license terms apply to all Open Publication works, unless otherwise explicitly stated in the document.

Mere aggregation of Open Publication works or a portion of an Open Publication work with other works or programs on the same media shall not cause this license to apply to those other works. The aggregate work shall contain a notice specifying the inclusion of the Open Publication material and appropriate copyright notice.

SEVERABILITY. If any part of this license is found to be unenforceable in any jurisdiction, the remaining portions of the license remain in force.

NO WARRANTY. Open Publication works are licensed and provided "as is" without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement.

IV. REQUIREMENTS ON MODIFIED WORKS

All modified versions of documents covered by this license, including translations, anthologies, compilations and partial documents, must meet the following requirements:

1. The modified version must be labeled as such.
2. The person making the modifications must be identified and the modifications dated.
3. Acknowledgement of the original author and publisher if applicable must be retained according to normal academic citation practices.
4. The location of the original unmodified document must be identified.
5. The original author's (or authors') name(s) may not be used to assert or imply endorsement of the resulting document without the original author's (or authors') permission.

V. GOOD-PRACTICE RECOMMENDATIONS

In addition to the requirements of this license, it is requested from and strongly recommended of redistributors that:

1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.
2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.

Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy and CD-ROM expression of an Open Publication-licensed work to its author(s).

VI. LICENSE OPTIONS

The author(s) and/or publisher of an Open Publication-licensed document may elect certain options by appending language to the reference to or copy of the license. These options are considered part of the license instance and must be included with the license (or its incorporation by reference) in derived works.

A. To prohibit distribution of substantively modified versions without the explicit permission of the author(s). "Substantive modification" is defined as a change to the semantic content of the document, and excludes mere changes in format or typographical corrections.

To accomplish this, add the phrase 'Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.' to the license reference or copy.

B. To prohibit any publication of this work or derivative works in whole or in part in standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

To accomplish this, add the phrase 'Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.' to the license reference or copy.

OPEN PUBLICATION POLICY APPENDIX:

(This is not considered part of the license.)

Open Publication works are available in source format via the Open Publication home page at http://works.opencontent.org/.

Open Publication authors who want to include their own license on Open Publication works may do so, as long as their terms are not more restrictive than the Open Publication license.

If you have questions about the Open Publication License, please contact TBD, and/or the Open Publication Authors' List at opal@opencontent.org, via email.