

# A Brief RPM Guide

Jim Perrin

January 14 2003

## 1 RPM Overview

The RPM Package Manager (RPM) is a powerful command line driven package management system capable of installing, uninstalling, verifying, querying, and updating computer software packages. Each software package consists of an archive of files along with information about the package like its version, a description, and the like. There is also a related API ("Application Program Interface"), permitting advanced developers to bypass 'shelling out' to a command line, and to manage such transactions from within a native coding language. RPM has become the defacto format for LSB compliant distributions.

### Basic RPM Commands

1. -U Update or install the chosen package. This can be done via http or ftp in addition to using local files.
2. -i Install the selected package, allows for multiple versions.
3. -e Remove. This allows you to remove the packages you specify.
4. -v Verbose. Print out more information than usual.
5. -F Freshen. This is similar to -U but will only work on if an earlier version of the package is already installed.
6. -h Hash. This makes rpm pretty when you install packages, and gives you a progress bar.
7. -qa Query all. This allows you to query for every package installed on the system. You can limit the search by giving it a value like `rpm -qa "send*"`
8. -ql lists the files associated with a package.
9. -qg lists all installed packages in a specific group, for example "System Environment/Shells".
10. -qi lists the spec file header for the given package. This is one of the things that makes rpm useful for automation. We'll get into this later.
11. --force DO NOT EVER DO THIS. This is a last resort command, and should only be used by functionally impaired admins. Unless you want to jack up your system, pretend this doesn't exist.

These options allow for the basics of package management. With them you can install, remove and update packages on your system. This by itself isn't really that great, so let's get into the meat of RPM.

### Advanced RPM Commands

These functions are more useful, but dig a little deeper into rpm and other applications

1. -V verifies packages on your system, and will inform you if a file has been altered, if the md5sum changes, if files are missing etc.
2. -K or --checksig verifies that the rpm was signed and verified either by gpg, pgp, or md5. This does require that you import the keys supplied by your rpm vendor.
3. --sign signs an rpm that you make, if you're so inclined.

4. `--rollback` allows you to roll back package upgrades for a specific time period. `rpm --rollback "1 week ago"` is a good example of this.
5. `--repackage` will recreate an rpm from files installed on your system. This is not a complete package, and must be rebuilt with `rpmbuild --rebuild`.

For the Gentoo crowd, you can do other various searches of the database to look at various statistics about the rpms on your system. `rpm -qa --qf "%{PLATFORM}\n"` This example will show you how many rpms you have installed on your system, and for which architectures. You can do similar searches for NAME, VERSION, SIZE, SOURCERPM, etc. It's thrilling I know.

## RPM Database Manipulation

The only thing I'm going to show you here is how to back up your rpm database. If you're doing more to it than this, you know more than the audience I'm writing for.

```
cd /var/lib;
tar czvf rpm.tar.gz rpm/;
```

You are now free to screw up your database any way you see fit.

## 2 Building with RPM

This is where just a little work on your part can save hours of troubleshooting later. If you use an rpm based distribution, installing anything from source without making it into an rpm is just lazy, and it's going to cause you problems down the road. Many times you will quickly get into a snowball effect installing from source and rpm on the same system. Dependencies will begin to surface and you will be reduced to using `--force` to get packages to install, which will render many rpm tools useless. Get in the good habit of making an rpm for a package if one does not exist already.

Many programs being delivered in source form are now including spec files which allows you to quickly compile them for your system, and install them properly all at the same time. To build source rpms, or other source packages, there are a few things you need to configure. The first thing to set up is a build environment for RPM to use. Most RPM based distros have a build environment set up for the root user, but it's generally a better idea not to build as root. To set up your build root you need to have two things, a `.rpmmacros` file, and a build directory structure. Below is my `.rpmmacros` file.

```
%_topdir /home/evo/build
%debug_package %{nil}
%_unpackaged_files_terminate_build 0
```

The first line tells RPM what directory to use for builds and storage. The second tells rpm not to build the debug packages. This is entirely personal preference. I don't build them because they tend to be huge, and I don't do much debugging. The last line tells RPM not to worry if the build has files that I chose not to use in the RPM. The next thing to do is to actually make the directory structure RPM will use. The simplest way is to `mkdir ~/build ; cp -r /usr/src/redhat/* ~/build`

## Spec File Structure

A very basic spec file has a pretty simple layout, and should be easy to follow. Below is a sample spec file for a side project. It should provide a clean simple look to start from.

```
Summary: A searchable xml/html based mailinglist archiver
Name: lurker
Version: 1.1
Release: 1.evo.2
Group: Applications/Internet
License: GPL
URL: http://lurker.sourceforge.net/
Packager: Jim Perrin <perrin@ohio.edu>
```

```

Source0: lurker-%{version}.tar.gz
Source1: mimelib-3.1.1.tar.gz

BuildRequires: zlib-devel
Requires: libxslt
BuildRoot: %{_tmppath}/%{name}-%{version}-root

%description
Lurker is not just another mailing list archiver. It is capable of
handling gigabytes of mail without slowing down. Lurker has been
designed to scale to support sites with thousands of concurrent users
and hundreds of new messages a second. If you run a high-volume
mailing list archive, you should seriously consider lurker for this
alone.

%prep
%setup -a 1

%build
./configure --with-mimelib-local --prefix=$RPM_BUILD_ROOT --localstatedir=$RPM_BUILD_ROOT/
make %{?_smp_mflags}

%install
%makeinstall
install -d -m0755 $RPM_BUILD_ROOT/%{_sysconfdir}
install -m0755 lurker.conf $RPM_BUILD_ROOT/%{_sysconfdir}/lurker.conf
install -d -m2775 $RPM_BUILD_ROOT/%{_localstatedir}/www/lurkdb
%clean
rm -rf $RPM_BUILD_ROOT

%files
%defattr(-,root,root)
%doc AUTHORS ChangeLog COPYING FAQ NEWS README INSTALL
%{_mandir}/man1/*
%{_bindir}/*
%{_libdir}/*
%config(noreplace) %{_sysconfdir}/lurker.conf
%defattr(-,apache,apache)
%{_localstatedir}/www/lurker/
%{_localstatedir}/www/lurkdb/

%changelog

* Fri Jan 2 2004 Jim Perrin <perrin@ohio.edu>
- Changed the spec file to better use macros

```

The one thing this spec file doesn't show is how to patch source using the rpm build process. This will be covered during the presentation.

## Spec macros and scripting

If you've noticed, there are several sections of the spec file such as %prep and %setup that take care of the more mundane aspects of building from source. The list below contains some basic macros for use with RPM. Using a macro is the preferred method of packaging, because it allows for the most flexibility.

1. %setup unpacks source files and creates the buildroot.

2. %configure runs ./configure with standard options. It expands to the listing below on redhat systems. For other RPM systems, you can test this with rpm -eval %configure

```
./configure --host=i686-pc-linux-gnu --build=i686-pc-linux-gnu \  
--target=i386-redhat-linux-gnu \  
--program-prefix= \  
--prefix=/usr \  
--exec-prefix=/usr \  
--bindir=/usr/bin \  
--sbindir=/usr/sbin \  
--sysconfdir=/etc \  
--datadir=/usr/share \  
--includedir=/usr/include \  
--libdir=/usr/lib \  
--libexecdir=/usr/libexec \  
--localstatedir=/var \  
--sharedstatedir=/usr/com \  
--mandir=/usr/share/man \  
--infodir=/usr/share/info
```

3. %make runs (get this) "make".
4. %makeinstall runs "make install"
5. The most useful macros are used for file positioning, such as %{\_bindir}.

Once you've got your spec file set up, copy or move the source to the SOURCES directory, and put your spec file in the SPECS directory. Once this is done you can issue `rpmbuild -ba foo.spec` and the build process will begin. If you just want to rebuild a source rpm (src.rpm) without making changes to it you can issue `rpmbuild --rebuild foo.src.rpm`. If for some reason you want to build a architecture specific package (a kernel rpm is a good reason) you can use `rpmbuild --rebuild --target=i686 foo.src.rpm`.

## File Control and Attributes

This is actually where you install the files and set the permissions for them. The %files section lists the files associated with a given rpm, while the %defattr sets the owner and permissions for those files. One important thing to watch out for is assigning a specific user in an rpm. If your rpm is to be used on more than one distribution, you will run into users that don't exist, and rpm will do mean things.

## 3 Sources

There are several good RPM sources available online. A few of them are listed below. While many of the sites supply rpms on their own, they also show spec files of the packages they have, which makes it a great place to get pointers for how things should be done.

1. <http://freshrpms.net> - A good source for 3rd party applications. Most rpm tools include this site as a standard resource.
2. <http://dag.wieers.com/home-made/apt/packages.php> - This site has a few harder to find packages than freshrpms, and does a nice job of version control.
3. <http://rpm.org> - This site is the RPM homepage, and has some good how-to pointers, but they may be a bit dated. Still a worthy resource. Look closely at their maximum rpm information.
4. [#rpm](irc://freenode.net) - The irc channel where a few rpm people give pointers. Make sure you've exhausted other resources before you go pestering the channel, some people can get quite touchy.