

**NAME**

rpm – Red Hat Package Manager

**SYNOPSIS**

**rpm** [options]

**DESCRIPTION**

**rpm** is a powerful *package manager*, which can be used to build, install, query, verify, update, and uninstall individual software packages. A *package* consists of an archive of files, and package information, including name, version, and description.

One of the following basic modes must be selected: *Initialize Database, Rebuild Database, Build Package, Recompile Package, Build Package from Tarball, Query, Show Querytags, Install, Freshen, Uninstall, Verify, Signature Check, Resign, Add Signature, set owners and groups* and *Show Configuration*.

Database maintenance:

**rpm -i [--initdb]**  
**rpm -i [--rebuilddb]**

Building:

**rpm [-b[t] [package\_spec]+**  
**rpm [--rebuild] [sourcerpm]+**  
**rpm [--tarbuild] [tarredsource]+**

Querying:

**rpm [--query] [queryoptions]**  
**rpm [--querytags]**

Maintaining installed packages:

**rpm [--install] [installoptions] [package\_file]+**  
**rpm [--freshen|-F] [installoptions] [package\_file]+**  
**rpm [--uninstall|-e] [uninstalloptions] [package]+**  
**rpm [--verify|-V] [verifyoptions] [package]+**

Signatures:

**rpm [--verify|-V] [verifyoptions] [package]+**  
**rpm [--resign] [package\_file]+**  
**rpm [--addsign] [package\_file]+**

Miscellaneous:

**rpm [--showrc]**  
**rpm [--setperms] [package]+**  
**rpm [--setgids] [package]+**

**GENERAL OPTIONS**

These options can be used in all the different modes.

**-vv** Print lots of ugly debugging information.

**--quiet**

Print as little as possible – normally only error messages will be displayed.

**--help** Print a longer usage message then normal.

**--version**

Print a single line containing the version number of **rpm** being used.

**--rcfile** *<filelist>*

Each of the files in the colon separated *<filelist>* is read sequentially by **rpm** for configuration information. The default *<filelist>* is `/usr/lib/rpm/rpmrc:/etc/rpmrc:~/.rpmrc`. Only the first file in the list must exist, and tildes will be expanded to the value of **\$HOME**.

**--root** *<dir>*

Use the system rooted at *<dir>* for all operations. Note that this means the database will be read or modified under *<dir>* and any *pre* or *post* scripts are run after a `chroot()` to *<dir>*.

**--dbpath** *<path>*

Use RPM database in *<path>*.

**--justdb**

Update only the database, not the filesystem.

**--ftpproxy** *<host>*, **--httpproxy** *<host>*

Use *<host>* as an FTP or HTTP proxy host. See **FTP/HTTP OPTIONS**.

**--ftpport** *<port>*, **--httpport** *<port>*

Use *<port>* as the FTP or HTTP port on the proxy host. See **FTP/HTTP OPTIONS**.

**--pipe** *<cmd>*

Pipes the output of **rpm** to the command *<cmd>*.

**INSTALL AND UPGRADE OPTIONS**

The general form of an rpm install command is

```
rpm -i [install-options] <package_file>+
```

This installs a new package. The general form of an rpm upgrade command is

```
rpm -U [install-options] <package_file>+
```

This upgrades or installs the package currently installed to the version in the new RPM. This is the same as install, except all other version of the package are removed from the system.

```
rpm [-F|--freshen] [install-options] <package_file>+
```

This will upgrade packages, but only if an earlier version currently exists.

The *<package\_file>* may be specified as an ftp or http URL, in which case the package will be downloaded before being installed. See **FTP/HTTP OPTIONS** for information on RPM's built-in ftp and http support.

**--force**

Same as using **--replacepkgs**, **--replacefiles**, and **--oldpackage**.

**-h, --hash**

Print 50 hash marks as the package archive is unpacked. Use with **-v** for a nice display.

**--oldpackage**

Allow an upgrade to replace a newer package with an older one.

**--percent**

Print percentages as files are unpacked from the package archive. This is intended to make RPM easy to run from other tools.

**--replacefiles**

Install the packages even if they replace files from other, already installed, packages.

- replacepkgs**  
Install the packages even if some of them are already installed on this system.
- allfiles**  
Installs or upgrades all the missing files in the package, regardless if they exist.
- nodeps**  
Don't do a dependency check before installing or upgrading a package.
- noscripts**  
Don't execute the preinstall or postinstall scripts.
- notriggers**  
Don't execute scripts which are triggered by the installation of this package.
- ignoresize**  
Don't check mount file systems for sufficient disk space before installing this package.
- excludepath** *<path>*  
Don't install files whose name begins with *<path>*.
- excludedocs**  
Don't install any files which are marked as documentation (which includes man pages and texinfo documents).
- includedocs**  
Install documentation files. This is the default behavior.
- test** Do not install the package, simply check for and report potential conflicts.
- ignorearch**  
This allows installation or upgrading even if the architectures of the binary RPM and host don't match.
- ignoreos**  
This allows installation or upgrading even if the operating systems of the binary RPM and host don't match.
- prefix** *<path>*  
This sets the installation prefix to *<path>* for relocatable packages.
- relocate** *<oldpath>=<newpath>*  
For relocatable packages, translates the files that would be put in *<oldpath>* to *<newpath>*.
- badreloc**  
To be used in conjunction with **--relocate**, this forces the relocation even if the package isn't relocatable.
- noorder**  
Don't reorder the packages for an install. The list of packages would normally be reordered to satisfy dependencies.

## QUERY OPTIONS

The general form of an rpm query command is

**rpm -q [query-options]**

You may specify the format that package information should be printed in. To do this, you use the **[--queryformat]-qf** option, followed by the format string.

Query formats are modified versions of the standard **printf(3)** formatting. The format is made up of static strings (which may include standard C character escapes for newlines, tabs, and other special characters) and **printf(3)** type formatters. As **rpm** already knows the type to print, the type specifier must be omitted

however, and replaced by the name of the header tag to be printed, enclosed by {} characters. The **RPM-TAG\_** portion of the tag name may be omitted.

Alternate output formats may be requested by following the tag with *:typetag*. Currently, the following types are supported: **octal**, **date**, **shescape**, **perms**, **fflags**, and **depflags**.

For example, to print only the names of the packages queried, you could use **%{NAME}** as the format string. To print the packages name and distribution information in two columns, you could use **%-30{NAME}%{DISTRIBUTION}**.

**rpm** will print a list of all of the tags it knows about when it is invoked with the **--querytags** argument.

There are two subsets of options for querying: package selection, and information selection.

Package selection options:

*<package\_name>*

Query installed package named *<package\_name>*.

**-a, --all**

Query all installed packages

**--whatrequires** *<capability>*

Query all packages that requires *<capability>* for proper functioning.

**--whatprovides** *<virtual>*

Query all packages that provide the *<virtual>* capability.

**-f** *<file>*, **--file** *<file>*

Query package owning *<file>*.

**-g** *<group>*, **--group** *<group>*

Query packages with the group of *<group>*.

**-p** *<package\_file>*

Query an (uninstalled) package *<package\_file>*. The *<package\_file>* may be specified as an ftp or http style URL, in which case the package header will be downloaded and queried. See **FTP/HTTP OPTIONS** for information on RPM's built-in ftp and http client support.

**--specfile** *<specfile>*

Parse and query *<specfile>* as if it were a package. Although not all the information (e.g. file lists) is available, this type of query permits rpm to be used to extract information from spec files without having to write a specfile parser.

**--querybynumber** *<num>*

Query the *<num>*th database entry directly; this is helpful for debugging purposes.

**--triggeredby** *<pkg>*

Query packages that are triggered by packages *<pkg>*.

Information selection options:

**-i** Display package information, including name, version, and description. This uses the **--queryformat** if one was specified.

**-R, --requires**

List packages on which this package depends.

**--provides**

List capabilities this package provides.

- changelog**  
Display change information for the package.
- l, --list**  
List files in package.
- s, --state**  
Display the *states* of files in the package (implies **-l**). The state of each file is either *normal*, *not installed*, or *replaced*.
- d, --docfiles**  
List only documentation files (implies **-l**).
- c, --configfiles**  
List only configuration files (implies **-l**).
- scripts**  
List the package specific shell scripts that are used as part of the installation and uninstallation processes, if there are any.
- triggers, --triggerscripts**  
Display the trigger scripts, if any, which are contained in the package.
- dump**  
Dump file information as follows: path size mtime md5sum mode owner group isconfig isdoc rdev symlink. This must be used with at least one of **-l**, **-c**, **-d**.
- last** Orders the package listing by install time such that the latest packages are at the top.
- filesbypkg**  
This lists all the files in each package.
- triggerscripts**  
Shows all the trigger scripts for the selected packages.

## VERIFY OPTIONS

The general form of an rpm verify command is

```
rpm -V|-y|--verify [verify-options]
```

Verifying a package compares information about the installed files in the package with information about the files taken from the original package and stored in the rpm database. Among other things, verifying compares the size, MD5 sum, permissions, type, owner and group of each file. Any discrepancies are displayed. The package specification options are the same as for package querying.

Files that were not installed from the package, for example documentation files excluded on installation using the "**--excludedocs**" option, will be silently ignored.

Options that can be used in verify mode:

- nofiles**  
Ignores missing files when verifying.
- nomd5**  
Ignores MD5 checksum errors when verifying.
- nopgp**  
Ignores PGP checking errors when verifying.
- nofiles**  
Ignores missing files when verifying.

The format of the output is a string of 8 characters, a possible "c" denoting a configuration file, and then the file name. Each of the 8 characters denotes the result of a comparison of one attribute of the file to the value of that attribute recorded in the RPM database. A single "." (period) means the test passed. The following characters denote failure of certain tests:

<b>5</b>	MD5 sum
<b>S</b>	File size
<b>L</b>	Symlink
<b>T</b>	Mtime
<b>D</b>	Device
<b>U</b>	User
<b>G</b>	Group
<b>M</b>	Mode (includes permissions and file type)

## SIGNATURE CHECKING

The general form of an rpm signature check command is

```
rpm --checksig <package_file>+
```

This checks the PGP signature of package <package\_file> to ensure its integrity and origin. PGP configuration information is read from configuration files. See the section on PGP SIGNATURES for details.

## UNINSTALL OPTIONS

The general form of an rpm uninstall command is

```
rpm -e <package_name>+
```

### **--allmatches**

Remove all versions of the package which match <package\_name>. Normally an error is issued if <package\_name> matches multiple packages.

### **--noscripts**

Don't execute the preuninstall or postuninstall scripts.

### **--notriggers**

Don't execute scripts which are triggered by the removal of this package.

### **--nodeps**

Don't check dependencies before uninstalling the packages.

**--test** Don't really uninstall anything, just go through the motions. Useful in conjunction with the **-vv** option.

## BUILD OPTIONS

The general form of an rpm build command is

```
rpm -[b|t]O [build-options] <package_spec>+
```

The argument used is **-b** if a spec file is being used to build the package and **-t** if **RPM** should look inside of a gzipped (or compressed) tar file for the spec file to use. After the first argument, the next argument (*O*) specifies the stages of building and packaging to be done and is one of:

**-bp** Executes the "%prep" stage from the spec file. Normally this involves unpacking the sources and applying any patches.

- bl** Do a "list check". The "%files" section from the spec file is macro expanded, and checks are made to verify that each file exists.
- bc** Do the "%build" stage from the spec file (after doing the prep stage). This generally involves the equivalent of a "make".
- bi** Do the "%install" stage from the spec file (after doing the prep and build stages). This generally involves the equivalent of a "make install".
- bb** Build a binary package (after doing the prep, build, and install stages).
- bs** Build just the source package (after doing the prep, build, and install stages).
- ba** Build binary and source packages (after doing the prep, build, and install stages).

The following options may also be used:

- short-circuit**  
Skip straight to specified stage (ie, skip all stages leading up to the specified stage). Only valid with **-bc** and **-bi**.
- timecheck**  
Set the "timecheck" age (0 to disable). This value can also be configured by defining the macro "\_timecheck". The timecheck value expresses, in seconds, the maximum age of a file being packaged. Warnings will be printed for all files beyond the timecheck age.
- clean**  
Remove the build tree after the packages are made.
- rmsource**  
Remove the sources and spec file after the build (may also be used standalone, eg. "**rpm --rmsource foo.spec**").
- test** Do not execute any build stages. Useful for testing out spec files.
- sign** Embed a PGP signature in the package. This signature can be used to verify the integrity and the origin of the package. See the section on PGP SIGNATURES for configuration details.
- buildroot <dir>**  
When building the package, override the BuildRoot tag with directory <dir>.
- target <platform>**  
When building the package, interpret <platform> as **arch-vendor-os** and set the macros **\_target**, **\_target\_arch** and **\_target\_os** accordingly.
- buildarch <arch>**  
When building the package, set the architecture to <arch>. This option has been obsoleted by **--target** in *RPM 3.0*.
- buildos <os>**  
When building the package, set the architecture to <os>. This option has been obsoleted by **--target** in *RPM 3.0*.

## REBUILD AND RECOMPILE OPTIONS

There are two other ways to invoke building with rpm:

**rpm --recompile <source\_package\_file>+**

**rpm --rebuild <source\_package\_file>+**

When invoked this way, **rpm** installs the named source package, and does a prep, compile and install. In addition, **--rebuild** builds a new binary package. When the build has completed, the build directory is removed (as in **--clean**) and the the sources and spec file for the package are removed.

## SIGNING AN EXISTING RPM

**rpm --resign** *<binary\_package\_file>+*

This option generates and inserts new signatures for the listed packages. Any existing signatures are removed.

**rpm --addsign** *<binary\_package\_file>+*

This option generates and appends new signatures for the listed packages to those that already exist.

## PGP SIGNATURES

In order to use the signature feature RPM must be configured to run PGP, and it must be able to find a public key ring with RPM public keys in it. By default, RPM uses the PGP defaults to find the keyrings (honoring PGPPATH). If your key rings are not located where PGP expects them to be, you will need to configure the macro

**\_\_pgp\_path**

to be the location of the PGP key rings to use.

If you want to be able to sign packages you create yourself, you also need to create your own public and secret key pair (see the PGP manual). You will also need to configure the macros

**\_\_signature**

The signature type. Right now only pgp is supported.

**\_\_pgp\_name**

The name of the "user" whose key you wish to use to sign your packages.

When building packages you then add **--sign** to the command line. You will be prompted for your pass phrase, and your package will be built and signed.

For example, to be able to use PGP to sign packages as the user "John Doe <jdoe@foo.com>" from the key rings located in **/etc/rpm/.pgp** using the executable **/usr/bin/pgp** you would include

**%\_signature pgp**

**%\_pgp\_path /etc/rpm/.pgp**

**%\_pgp\_name John Doe <jdoe@foo.com>**

**%\_pgpbin /usr/bin/pgp**

in a macro configuration file. Use **/etc/rpm/macros** for per-system configuration and **~/rpmmacros** for per-user configuration.

## REBUILD DATABASE OPTIONS

The general form of an rpm rebuild database command is

**rpm --rebuilddb**

To rebuild a new database, do:

**rpm --initdb**

The only options for these modes are **--dbpath** and **--root**.

## SHOWRC

Running

**rpm --showrc**

shows the values RPM will use for all of the options that may be set in *rpmrc* files.

## FTP/HTTP OPTIONS

RPM includes simple FTP and HTTP clients to simplify installing and querying packages which are available over the internet. Package files for install, upgrade, and query operations may be specified as an ftp or http style URL:

**ftp://<user>:<password>@hostname:<port>/path/to/package.rpm**

If the **:password** portion is omitted, the password will be prompted for (once per user/hostname pair). If both the user and password are omitted, anonymous ftp is used. In all cases passive (PASV) ftp transfers are used.

RPM allows the following options to be used with ftp URLs:

**--ftpproxy** <hostname>

The host <hostname> will be used as a proxy server for all ftp transfers, which allows users to ftp through firewall machines which use proxy systems. This option may also be specified by configuring the macro **\_ftpproxy**.

**--ftpport** <port>

The TCP <port> number to use for the ftp connection on the proxy ftp server instead of the default port. This option may also be specified by configuring the macro **\_ftpport**.

RPM allows the following options to be used with http URLs:

**--httpproxy** <hostname>

The host <hostname> will be used as a proxy server for all http transfers. This option may also be specified by configuring the macro **\_httpproxy**.

**--httpport** <port>

The TCP <port> number to use for the http connection on the proxy http server instead of the default port. This option may also be specified by configuring the macro **\_httpport**.

## FILES

/usr/lib/rpm/rpmrc  
 /etc/rpmrc  
 ~/.rpmrc  
 /usr/lib/rpm/macros  
 /etc/rpm/macros  
 ~/.rpmmacros  
 /var/lib/rpm/conflictsindex.rpm  
 /var/lib/rpm/fileindex.rpm

/var/lib/rpm/groupindex.rpm  
/var/lib/rpm/nameindex.rpm  
/var/lib/rpm/packages.rpm  
/var/lib/rpm/providesindex.rpm  
/var/lib/rpm/requiredby.rpm  
/var/lib/rpm/triggerindex.rpm  
/tmp/rpm\*

**SEE ALSO**

*glint(8)*, *rpm2cpio(8)*, <http://www.rpm.org/>

**AUTHORS**

Marc Ewing <marc@redhat.com>

Jeff Johnson <jbj@redhat.com>

Erik Troan <ewt@redhat.com>