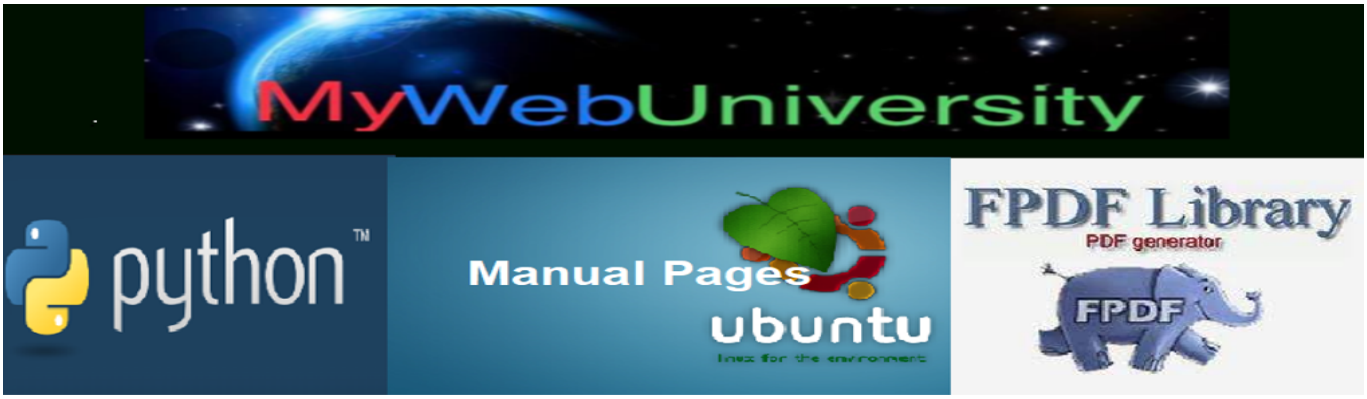

Rocky Enterprise Linux 9.2 Manual Pages on command 'rpmbuild.8'

$ man rpmbuild.8

RPMBUILD(8) RPMBUILD(8)

NAME

rpmbuild - Build RPM Package(s)

SYNOPSIS

BUILDING PACKAGES:

rpmbuild {-ba|-bb|-bp|-bc|-bi|-bl|-bs|-br} [rpmbuild-options] SPECFILE ...

rpmbuild {-ra|-rb|-rp|-rc|-ri|-rl|-rs|-rr} [rpmbuild-options] SOURCEPACKAGE ...

rpmbuild {-ta|-tb|-tp|-tc|-ti|-tl|-ts|-tr} [rpmbuild-options] TARBALL ...

rpmbuild {--rebuild|--recompile} SOURCEPKG ...

MISCELLANEOUS:

rpmbuild --showrc

rpmbuild-options

[--buildroot DIRECTORY] [--clean] [--nobuild] [--rmsource] [--rmspec] [--short-circuit]

[--build-in-place] [--noprep] [--noclean] [--nocheck] [--rpmfcdebug] [--target PLATFORM]

[--with OPTION] [--without OPTION]

DESCRIPTION

rpmbuild is used to build both binary and source software packages. A package consists of an archive of files and meta-data used to install and erase the archive files. The meta-data includes helper scripts, file attributes, and descriptive information about the pack? age. Packages come in two varieties: binary packages, used to encapsulate software to be installed, and source packages, containing the source code and recipe necessary to produce binary packages.

One of the following basic modes must be selected: Build Package, Build Package from Tar?

ball, Recompile Package, Show Configuration.

GENERAL OPTIONS

These options can be used in all the different modes.

-?, --help

Print a longer usage message then normal.

--version

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

--quiet

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

-v Print verbose information - normally routine progress messages will be displayed.

-vv Print lots of ugly debugging information.

--rpmfcdebug

Enables to debug dependencies generation.

--rcfile FILELIST

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

--pipe CMD

Pipes the output of rpm to the command CMD.

--dbpath DIRECTORY

Use the database in DIRECTORY rather than the default path /var/lib/rpm

--root DIRECTORY

Use the file system tree rooted at DIRECTORY for all operations. Note that this means the database within DIRECTORY will be used for dependency checks and any scriptlet(s) (e.g. %post if installing, or %prep if building, a package) will be run after a chroot(2) to DIRECTORY.

-D, --define='MACRO EXPR'

Defines MACRO with value EXPR.

--scm=SCM

Select the SCM to use with %autosetup, if one is not set in the spec file. Note that not all values for SCM, e.g., patch (the default) and gendiff, git, or quilt work interchangeably with all other patches and options stated in the %autosetup

line, especially option -pN.

BUILD OPTIONS

The general form of an rpm build command is

rpmbuild {-bSTAGE|-rSTAGE|-tSTAGE} [rpmbuild-options] FILE ...

The argument used is -b if a spec file is being used to build the package, -r if a source package is to be rebuilt and -t if rpmbuild should look inside of a (possibly compressed) tar file for the spec file to use.

Packages are built in a number of stages. The first six correspond to the following sections in a spec file: %prep, %generate_buildrequires, %build, %install, %check and %clean. Finally, binary and source packages are created in an assembly stage.

The STAGE character specifies the stage to finish with (after doing all the stages preceding it), and is one of:

-ba Perform a full build - executes up to and including the assembly stage. In most cases, this is the option to choose.

-bb Build just the binary packages - executes up to and including the assembly stage, but without creating the source package.

-bp Unpack the sources and apply any patches - executes the %prep stage only.

-bc Compile the sources - executes up to and including the %build stage. This generally involves the equivalent of a "make".

-bi Install the binaries into the build root - executes up to and including the %check stage. This generally involves the equivalent of a "make install" and "make check".

-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.

-bs Build just the source package - skips straight to the assembly stage, without executing any of the preceding stages or creating binary packages.

-br Build just the source package, but also parse and include dynamic build dependencies - executes up to and including the %generate_buildrequires stage and then skips straight to the assembly stage, without creating binary packages. This command can be used to fully resolve dynamic build dependencies. See the DYNAMIC BUILD DEPENDENCIES section for details.

The following options may also be used:

--buildroot DIRECTORY

When building a package, override the BuildRoot tag with directory DIRECTORY.

--clean

Remove the build tree after the packages are made.

--nobuild

Do not execute any build stages. Useful for testing out spec files.

--noprep

Do not execute %prep build stage even if present in spec.

--noclean

Do not execute %clean build stage even if present in spec.

--nocheck

Do not execute %check build stage even if present in spec.

--nodebuginfo

Do not generate debuginfo packages.

--nodeps

Do not verify build dependencies.

--rmsource

Remove the sources after the build (may also be used standalone, e.g. "rpmbuild --rmsource foo.spec").

--rmspec

Remove the spec file after the build (may also be used standalone, eg. "rpmbuild --rmspec foo.spec").

--short-circuit

Skip straight to specified stage (i.e., skip all stages leading up to the specified stage). Only valid with -bc, -bi, and -bb. Useful for local testing only. Pack? ages built this way will be marked with an unsatisfiable dependency to prevent their accidental use.

--build-in-place

Build from locally checked out sources. Sets _builddir to current working directo? ry. Skips handling of -n and untar in the %setup and the deletion of the buildSub? dir.

--target PLATFORM

When building the package, interpret PLATFORM as arch-vendor-os and set the macros %_target, %_target_cpu, and %_target_os accordingly.

--with OPTION

Enable configure OPTION for build.

--without OPTION

Disable configure OPTION for build.

REBUILD AND RECOMPILE OPTIONS

There are two other ways to invoke building with rpm:

rpmbuild --rebuild|--recompile SOURCEPKG ...

When invoked this way, rpmbuild installs the named source package, and does a prep, com? pile 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.

These options are now superseded by the -r* options which allow much more fine control over what stages of the build to run.

DYNAMIC BUILD DEPENDENCIES

When the %generate_buildrequires stage runs and some of the newly generated BuildRequires are not satisfied, rpmbuild creates an intermediate source package ending in buil? dreqs.nosrc.rpm, which has the new BuildRequires, and exits with code 11. This package can then be used in place of the original source package to resolve and install the miss? ing build dependencies in the usual way, such as with dnf-builddep(8).

Multiple layers of dynamic build dependencies may exist in a spec file; the presence of specific BuildRequires on the system may yield new BuildRequires next time a build is per? formed with the same source package. The easiest way to ensure that all dynamic build de? pendencies are satisfied is to run the -br command, install the new dependencies of the buildreqs.nosrc.rpm package and repeat the whole procedure until rpmbuild no longer exits with code 11.

If the -br command is coupled with --nodeps, exit code 11 is always returned and a buil? dreqs.nosrc.rpm package is always created.

SHOWRC

The command

rpmbuild --showrc

shows the values rpmbuild will use for all of the options are currently set in rpmrc and macros configuration file(s).

FILES

rpmrc Configuration

/usr/lib/rpm/rpmrc

/usr/lib/rpm/<vendor>/rpmrc

/etc/rpmrc

~/.rpmrc

Macro Configuration

/usr/lib/rpm/macros

/usr/lib/rpm/<vendor>/macros

/etc/rpm/macros

~/.rpmmacros

Temporary

/var/tmp/rpm*

SEE ALSO

gendiff(1),

popt(3),

rpm(8),

rpm2cpio(8),

rpmkeys(8)

rpmspec(8),

rpmsign(8),

rpmbuild --help - as rpm supports customizing the options via popt aliases it's impossible to guarantee that what's described in the manual matches what's available.

http://www.rpm.org/ <URL:http://www.rpm.org/>

AUTHORS

Marc Ewing <marc@redhat.com>

Jeff Johnson <jbj@redhat.com>

Erik Troan <ewt@redhat.com>