



Linux Ubuntu 26.04 Manual Pages on command 'dpkg-buildpackage.1'

\$ man dpkg-buildpackage.1

dpkg-buildpackage(1) dpkg suite dpkg-buildpackage(1)

NAME

dpkg-buildpackage - build binary or source packages from sources

SYNOPSIS

dpkg-buildpackage [option...] [--] [filename.dsc|directory]

DESCRIPTION

dpkg-buildpackage is a program that automates the process of building a Debian package.

The filename.dsc and directory arguments are supported since dpkg 1.22.7. Their semantics are experimental.

It consists of the following steps:

1. It runs the preinit hook before reading any source file. If a .dsc file has been specified it unpacks it anew and changes directory to it, if a directory has been specified it changes directory to it, otherwise it expects the current directory to contain the source tree. It prepares the build environment by setting various environment variables (see ENVIRONMENT), runs the init hook, and calls dpkg-source --before-build (unless -T or --target has been used).
2. It checks that the build-dependencies and build-conflicts are satisfied (unless -d or --no-check-builddeps is specified).
3. If one or more specific targets have been selected with the -T or --target option, it calls those targets and stops here. Otherwise it runs the preclean hook and calls fakeroot debian/rules clean to clean the build-tree (unless -nc or --no-pre-clean is specified).
4. It runs the source hook and calls dpkg-source -b to generate the source package (if a

source build has been requested with --build or equivalent options, and if no .dsc has been specified).

5. It runs the build hook and calls debian/rules build-target, then runs the binary hook followed by fakeroot debian/rules binary-target (unless a source-only build has been requested with --build=source or equivalent options). Note that build-target and binary-target are either build and binary (default case, or if an any and all build has been requested with --build or equivalent options), or build-arch and binary-arch (if an any and not all build has been requested with --build or equivalent options), or build-indep and binary-indep (if an all and not any build has been requested with --build or equivalent options).
6. It runs the buildinfo hook and calls dpkg-genbuildinfo to generate a .buildinfo file. Several dpkg-buildpackage options are forwarded to dpkg-genbuildinfo. If a .dsc has been specified, then it will be referenced in the generated .buildinfo file, as we can ascertain the provenance of the source tree.
7. It runs the changes hook and calls dpkg-genchanges to generate a .changes file. The name of the .changes file will depend on the type of build and will be as specific as necessary but not more; the name will be:

source-name_binary-version_arch.changes

for a build that includes any

source-name_binary-version_all.changes

otherwise for a build that includes all

source-name_source-version_source.changes.

otherwise for a build that includes source

Many dpkg-buildpackage options are forwarded to dpkg-genchanges.
8. It runs the postclean hook and if -tc or --post-clean is specified, it will call fakeroot debian/rules clean again.
9. It calls dpkg-source --after-build.
10. It runs the check hook and calls a package checker for the .changes file (if a command is specified in DEB_CHECK_COMMAND or with --check-command).
11. It runs the sign hook and signs using the OpenPGP backend (as long as it is not an UNRELEASED build, or --no-sign is specified) to sign the .dsc file (if any, unless -us or --unsigned-source is specified), the .buildinfo file (unless -ui, --unsigned-buildinfo, -uc or --unsigned-changes is specified) and the .changes file

(unless -uc or --unsigned-changes is specified).

12. If a .dsc file has been specified, it removes the extracted source directory.

13. It runs the done hook.

OPTIONS

All long options can be specified both on the command line and in the dpkg-buildpackage system and user configuration files. Each line in the configuration file is either an option (exactly the same as the command line option but without leading hyphens) or a comment (if it starts with a #?).

--build=type

Specifies the build type from a comma-separated list of components (since dpkg 1.18.5).

All the specified components get combined to select the single build type to use, which implies a single build run with a single .changes file generated. Passed to dpkg-genchanges.

The allowed values are:

source

Builds the source package.

Note: When using this value standalone and if what you want is simply to (re-)build the source package from a clean source tree, using dpkg-source directly is always a better option as it does not require any build dependencies to be installed which are otherwise needed to be able to call the clean target.

any Builds the architecture specific binary packages.

all Builds the architecture independent binary packages.

binary

Builds the architecture specific and independent binary packages. This is an alias for any,all.

full

Builds everything. This is an alias for source,any,all, and the same as the default case when no build option is specified.

-g Equivalent to --build=source,all (since dpkg 1.17.11).

-G Equivalent to --build=source,any (since dpkg 1.17.11).

-b Equivalent to --build=binary or --build=any,all.

-B Equivalent to --build=any.

-A Equivalent to --build=all.

-S Equivalent to --build=source.

-F Equivalent to --build=full, --build=source,binary or --build=source,any,all (since dpkg 1.15.8).

--target=target[,...]

--target target[,...]

-T, --rules-target=target[,...]

Calls debian/rules target once per target specified, after having setup the build environment (except for calling dpkg-source --before-build), and stops the package build process here (since dpkg 1.15.0, long option since dpkg 1.18.8, multi-target support since dpkg 1.18.16). If --as-root is also given, then the command is executed as root (see --root-command). Note that known targets that are required to be run as root do not need this option (i.e. the clean, binary, binary-arch and binary-indep targets).

--as-root

Only meaningful together with --target (since dpkg 1.15.0). Requires that the target be run with root rights.

-si

-sa

-sd

-vversion

-Cchanges-description

-mmaintainer-address

-emaintainer-address

Passed unchanged to dpkg-genchanges. See its manual page.

--build-by=maintainer-address

--source-by=maintainer-address (since dpkg 1.21.10)

Pass as -m to dpkg-genchanges. See its manual page.

--release-by=maintainer-address

--changed-by=maintainer-address (since dpkg 1.21.10)

Pass as -e to dpkg-genchanges. See its manual page.

-a, --host-arch architecture

Specify the Debian architecture we build for (long option since dpkg 1.17.17). The architecture of the machine we build on is determined automatically, and is also the default for the host machine.

`-t, --host-type gnu-system-type`

Specify the GNU system type we build for (long option since dpkg 1.17.17). It can be used in place of `--host-arch` or as a complement to override the default GNU system type of the host Debian architecture.

`--target-arch architecture`

Specify the Debian architecture the binaries built will build for (since dpkg 1.17.17).

The default value is the host machine.

`--target-type gnu-system-type`

Specify the GNU system type the binaries built will build for (since dpkg 1.17.17). It can be used in place of `--target-arch` or as a complement to override the default GNU system type of the target Debian architecture.

`-P, --build-profiles=profile[,...]`

Specify the profile(s) we build, as a comma-separated list (since dpkg 1.17.2, long option since dpkg 1.18.8). The default behavior is to build for no specific profile.

Also sets them (as a space separated list) as the `DEB_BUILD_PROFILES` environment variable which allows, for example, `debian/rules` files to use this information for conditional builds.

`-j, --jobs[=jobs|auto]`

Specifies the number of jobs allowed to be run simultaneously (since dpkg 1.14.7, long option since dpkg 1.18.8). The number of jobs matching the number of online processors if `auto` is specified (since dpkg 1.17.10), or unlimited number if `jobs` is not specified.

The default behavior is `auto` (since dpkg 1.18.11) in non-forced mode (since dpkg 1.21.10), and as such it is always safer to use with any package including those that are not parallel-build safe. Setting the number of jobs to 1 will restore serial execution.

Will add `parallel=jobs` or `parallel` to the `DEB_BUILD_OPTIONS` environment variable which allows `debian/rules` files to opt-in to use this information for their own purposes. The `jobs` value will override the `parallel=jobs` or `parallel` option in the `DEB_BUILD_OPTIONS` environment variable. Note that the `auto` value will get replaced by the actual number of currently active processors, and as such will not get propagated to any child process. If the number of online processors cannot be inferred then the code will fallback to using serial execution (since dpkg 1.18.15), although this should only happen on exotic and unsupported systems.

`-J, --jobs-try[=jobs|auto]`

This option (since dpkg 1.18.2, long option since dpkg 1.18.8) is equivalent to the `-j` above.

Since the behavior for `-j` changed in dpkg 1.21.10 to the opt-in mode, you can use this option instead if you need to guarantee semantics across dpkg release series.

`--jobs-force[=jobs|auto]`

This option (since dpkg 1.21.10) is equivalent to the `--jobs` option except that it will enable forced parallel mode, by adding the `make -j` option with the computed number of parallel jobs to the `MAKEFLAGS` environment variable.

This should cause all subsequent make invocations to inherit the option, thus forcing the parallel setting on the packaging (and possibly the upstream build system if that uses `make(1)`) regardless of their support for parallel builds, which might cause build failures.

Note: Any Makefile that is not parallel-safe should be considered to be buggy. These should either be made parallel-safe, or marked as not being safe with the `make(1)` `.NOTPARALLEL` target.

`-D, --check-builddeps`

Check build dependencies and conflicts; abort if unsatisfied (long option since dpkg 1.18.8). This is the default behavior.

`-d, --no-check-builddeps`

Do not check build dependencies and conflicts (long option since dpkg 1.18.8).

`--ignore-builtin-builddeps`

Do not check built-in build dependencies and conflicts (since dpkg 1.18.2). These are the distribution specific implicit build dependencies usually required in a build environment, the so called Build-Essential package set.

`--rules-requires-root`

Do not honor the Rules-Requires-Root field, falling back to its legacy default value `binary-targets` (since dpkg 1.19.1).

`-nc, --no-pre-clean`

Do not clean the source tree before building (long option since dpkg 1.18.8). Implies `-b` if nothing else has been selected among `-F`, `-g`, `-G`, `-B`, `-A` or `-S`. Implies `-d` with `-S` (since dpkg 1.18.0).

`--pre-clean`

Clean the source tree before building (since dpkg 1.18.8). This is the default behavior.

`-tc, --post-clean`

Clean the source tree (using gain-root-command `debian/rules clean`) after the package has been built (long option since dpkg 1.18.8).

`--no-post-clean`

Do not clean the source tree after the package has been built (since dpkg 1.19.1). This is the default behavior.

`--sanitize-env`

Sanitize the build environment (since dpkg 1.20.0). This will reset or remove environment variables, umask, and any other process attributes that might otherwise adversely affect the build of packages. Because the official entry point to build packages is `debian/rules`, packages cannot rely on these settings being in place, and thus should work even when they are not. What to sanitize is vendor specific.

`-r, --root-command=gain-root-command`

When `dpkg-buildpackage` needs to execute part of the build process as root, it prefixes the command it executes with `gain-root-command` if one has been specified (long option since dpkg 1.18.8). Otherwise, if none has been specified, `fakeroot` will be used by default, if the command is present. `gain-root-command` should start with the name of a program on the `PATH` and will get as arguments the name of the real command to run and the arguments it should take. `gain-root-command` can include parameters (they must be space-separated) but no shell metacharacters. `gain-root-command` might typically be `fakeroot`, `sudo`, `super` or `really`. `su` is not suitable, since it can only invoke the user's shell with `-c` instead of passing arguments individually to the command to be run.

`-R, --rules-file=rules-file`

Building a Debian package usually involves invoking `debian/rules` as a command with several standard parameters (since dpkg 1.14.17, long option since dpkg 1.18.8). With this option it's possible to use another program invocation to build the package (it can include space separated parameters). Alternatively it can be used to execute the standard rules file with another make program (for example by using `/usr/local/bin/make -f debian/rules` as `rules-file`).

`--check-command=check-command`

Command used to check the `.changes` file itself and any artifact built referenced in the

file (since dpkg 1.17.6). The command should take the .changes pathname as an argument.

This command will usually be lintian.

--check-option=opt

Pass option opt to the check-command specified with DEB_CHECK_COMMAND or --check-command (since dpkg 1.17.6). Can be used multiple times.

--hook-hook-name=hook-command

Set the specified shell code hook-command as the hook hook-name, which will run at the times specified in the run steps (since dpkg 1.17.6). The hooks will always be executed even if the following action is not performed (except for the binary hook). All the hooks will run in the unpacked source directory.

Some hooks can receive addition information through environment variables (since dpkg 1.22.0). All hooks get the hook name in the DPKG_BUILDPACKAGE_HOOK_NAME environment variable (since dpkg 1.22.0).

Note: Hooks can affect the build process, and cause build failures if their commands fail, so watch out for unintended consequences.

The current hook-name supported are:

preinit

init

preclean

source

Gets DPKG_BUILDPACKAGE_HOOK_SOURCE_OPTIONS with the space-separated lists of options that will be passed to the dpkg-source call.

build

Gets DPKG_BUILDPACKAGE_HOOK_BUILD_TARGET with the name of the debian/rules build target called. Before dpkg 1.22.7 the variable was only set if the target was called.

binary

Gets DPKG_BUILDPACKAGE_HOOK_BINARY_TARGET with the name of the debian/rules binary target called, but only if called.

buildinfo

Gets DPKG_BUILDPACKAGE_HOOK_BUILDINFO_OPTIONS with the space-separated lists of options that will be passed to the dpkg-genbuildinfo call.

changes

Gets DPKG_BUILDPACKAGE_HOOK_CHANGES_OPTIONS with the space-separated lists of options that will be passed to the dpkg-genchanges call.

postclean

check

Gets DPKG_BUILDPACKAGE_HOOK_CHECK_OPTIONS with the space-separated lists of options that will be passed to the check command call.

sign

done

The hook-command supports the following substitution format string, which will be applied to it before execution:

%% A single % character.

%a A boolean value (0 or 1), representing whether the following action is being performed.

%p The source package name.

%v The source package version.

%s The source package version (without the epoch).

%u The upstream version.

--buildinfo-file=filename

Set the filename for the generated .buildinfo file (since dpkg 1.21.0).

--buildinfo-option=opt

Pass option opt to dpkg-genbuildinfo (since dpkg 1.18.11). Can be used multiple times.

--sign-backend=sign-backend

Specify an OpenPGP backend interface to use when invoking the sign-command (since dpkg 1.21.10).

The default is auto, where the best current backend available will be used. The specific OpenPGP backends supported in order of preference are:

sop Any conforming Stateless OpenPGP (SOP) implementation, such as sqop, rsop, gosop or pgpainless-cli.

sq From Sequoia-PGP.

gpg From GnuPG.

-p, --sign-command=sign-command

When dpkg-buildpackage needs to execute an OpenPGP backend command to sign a source control (.dsc) file, a .buildinfo file or a .changes file it will run sign-command

(searching the PATH if necessary) instead of the default or auto-detected backend command (long option since dpkg 1.18.8). sign-command will get all the backend specific arguments according to the --sign-backend selected. sign-command should not contain spaces or any other shell metacharacters.

-k, --sign-keyid=key-id

--sign-key=key-id

Specify an OpenPGP key-ID (either a fingerprint or a user-ID) for the secret key to use when signing packages (--sign-key since dpkg 1.18.8, --sign-keyid since dpkg 1.21.10).

--sign-keyfile=key-file

Specify an OpenPGP key-file containing the secret key to use when signing packages (since dpkg 1.21.10).

Note: For security reasons the key-file is best kept locked with a password.

-us, --unsigned-source

Do not sign the source package (long option since dpkg 1.18.8).

-ui, --unsigned-buildinfo

Do not sign the .buildinfo file (since dpkg 1.18.19).

-uc, --unsigned-changes

Do not sign the .buildinfo and .changes files (long option since dpkg 1.18.8).

--no-sign

Do not sign any file, this includes the source package, the .buildinfo file and the .changes file (since dpkg 1.18.20).

--force-sign

Force the signing of the resulting files (since dpkg 1.17.0), regardless of -us, --unsigned-source, -ui, --unsigned-buildinfo, -uc, --unsigned-changes or other internal heuristics.

-sn

-ss

-sA

-sk

-su

-sr

-sK

-sU

-sR

-i, --diff-ignore[=regex]

-I, --tar-ignore[=pattern]

-z, --compression-level=level

-Z, --compression=compressor

Passed unchanged to dpkg-source. See its manual page.

--source-option=opt

Pass option opt to dpkg-source (since dpkg 1.15.6). Can be used multiple times.

--changes-file=filename

Set the filename for the generated .changes file (since dpkg 1.21.0).

--changes-option=opt

Pass option opt to dpkg-genchanges (since dpkg 1.15.6). Can be used multiple times.

--admindir=dir

--admindir dir

Change the location of the dpkg database (since dpkg 1.14.0). The default location is /var/lib/dpkg.

-, --help

Show the usage message and exit.

--version

Show the version and exit.

ENVIRONMENT

External environment

DEB_CHECK_COMMAND

If set, it will be used as the command to check the .changes file (since dpkg 1.17.6).

Overridden by the --check-command option.

DEB_SIGN_KEYID

If set, it will be used to sign the .changes, .buildinfo and .dsc files (since dpkg 1.17.2). Overridden by the --sign-keyid option.

DEB_SIGN_KEYFILE

If set, it will be used to sign the .changes, .buildinfo and .dsc files (since dpkg 1.21.10). Overridden by the --sign-keyfile option.

DEB_BUILD_OPTIONS

If set, it will contain a space-separated list of options that affect the behavior of

some dpkg tools involved in package building, and might affect the package build process if the code in debian/rules honors them. These options can have parameters specified immediately after an equal sign (=). For options that support multiple parameters, these will not be separated by spaces, as these are reserved to separate options.

The following are the options known and supported by dpkg tools, other options honored by debian/rules might be defined by distribution specific policies.

parallel=N

The debian/rules in the packaging might use this option to set up the build process to use N parallel jobs. It is overridden by the --jobs and --jobs-force options.

nocheck

dpkg-buildpackage will ignore the DEB_CHECK_COMMAND variable. The debian/rules in the packaging is not expected to run test suites during the build.

noot

If debian/rules calls dpkg-buildflags to set up the build flags, those will be set to not enable any optimizations.

nostrip

The debian/rules in the packaging should ensure that objects do not get the debugging information stripped. If debian/rules includes the mk/buildtools.mk make fragment the STRIP make variable will respect this option.

terse

dpkg-buildpackage will append the --no-print-directory make(1) flag to the MAKEFLAGS environment variable. The debian/rules in the packaging should reduce verbosity, while not being completely quiet.

hardening=feature-spec

reproducible=feature-spec

abi=feature-spec

future=feature-spec

qa=feature-spec

optimize=feature-spec

sanitize=feature-spec

These are feature areas that control build flag features. See dpkg-buildflags(1) for further details.

If set, it will be used as the active build profile(s) for the package being built (since dpkg 1.17.2). It is a space separated list of profile names. Overridden by the -P option.

DPKG_COLORS

Sets the color mode (since dpkg 1.18.5). The currently accepted values are: auto (default), always and never.

DPKG_NLS

If set, it will be used to decide whether to activate Native Language Support, also known as internationalization (or i18n) support (since dpkg 1.19.0). The accepted values are: 0 and 1 (default).

Internal environment

Even if dpkg-buildpackage exports some variables, debian/rules should not rely on their presence and should instead use the respective interface to retrieve the needed values, because that file is the main entry point to build packages and running it standalone should be supported.

DEB_BUILD_*

DEB_HOST_*

DEB_TARGET_*

dpkg-architecture is called with the -a and -t parameters forwarded. Any variable that is output by its -s option is integrated in the build environment.

DEB_RULES_REQUIRES_ROOT

This variable is set to the value obtained from the Rules-Requires-Root field, the dpkg-build-api level or from the command-line. When set, it will be a valid value for the Rules-Requires-Root field. It is used to notify debian/rules whether the rootless-builds.txt specification is supported.

DEB_GAIN_ROOT_CMD

This variable is set to gain-root-command when the field Rules-Requires-Root is set to a value different to no and binary-targets.

SOURCE_DATE_EPOCH

This variable is set to the Unix timestamp since the epoch of the latest entry in debian/changelog, if it is not already defined.

FILES

/etc/dpkg/buildpackage.conf

System wide configuration file

\$XDG_CONFIG_HOME/dpkg/buildpackage.conf or

\$HOME/.config/dpkg/buildpackage.conf

User configuration file.

NOTES

Compiler flags are no longer exported

Between dpkg 1.14.17 and 1.16.1, dpkg-buildpackage exported compiler flags (CFLAGS, CXXFLAGS, FFLAGS, CPPFLAGS and LDFLAGS) with values as returned by dpkg-buildflags. This is no longer the case.

Default build targets

dpkg-buildpackage is using the build-arch and build-indep targets since dpkg 1.16.2. Before dpkg 1.22.7, there was code to try to detect the missing targets and fallback on the build target. Those targets are thus mandatory.

SECURITY

Building binary or source packages should only be performed over trusted source data.

BUGS

It should be possible to specify spaces and shell metacharacters and initial arguments for gain-root-command and sign-command.

SEE ALSO

/usr/share/doc/dpkg/spec/rootless-builds.txt, dpkg-source(1), dpkg-architecture(1), dpkg-buildflags(1), dpkg-genbuildinfo(1), dpkg-genchanges(1), fakeroot(1), lintian(1), SOP <<https://datatracker.ietf.org/doc/draft-dkg-openpgp-stateless-cli/>>, sq(1), gpg(1).

1.23.7

2026-03-31

dpkg-buildpackage(1)