dpkg(1) dpkg suite dpkg(1)

NAME

dpkg - package manager for Debian

SYNOPSIS

dpkg [option...] action

WARNING

This manual is intended for users wishing to understand dpkg's command line options and package states in more detail than that provided by dpkg --help.

It should not be used by package maintainers wishing to understand how dpkg will install their packages. The descriptions of what dpkg does when installing and removing packages are particularly inadequate.

DESCRIPTION

dpkg is a medium-level tool to install, build, remove and manage Debian packages. The primary and more user-friendly front-end for dpkg as a CLI (command-line interface) is apt(8) and as a TUI (terminal user interface) is aptitude(8). dpkg itself is controlled entirely via command line parameters, which consist of exactly one action and zero or more options. The action-parameter tells dpkg what to do and

dpkg can also be used as a front-end to dpkg-deb(1) and dpkg-query(1). The list of supported actions can be found later on in the ACTIONS section. If any such action is encountered dpkg just runs dpkg-deb or dpkg-query with the parameters given to it, but no specific options are currently passed to them, to use any such option the back-ends need to be called directly.

INFORMATION ABOUT PACKAGES

dpkg maintains some usable information about available packages. The information is divided in three classes: states, selection states and flags. These values are intended to be changed mainly with dselect.

Package states

not-installed

The package is not installed on your system.

config-files

Only the configuration files or the postrm script and the data it needs to remove of the package exist on the system.

half-installed

The installation of the package has been started, but not completed for some reason.

unpacked

The package is unpacked, but not configured.

half-configured

The package is unpacked and configuration has been started, but not yet completed for some reason.

triggers-awaited

The package awaits trigger processing by another package.

triggers-pending

The package has been triggered.

installed

The package is correctly unpacked and configured.

Package selection states

install

The package is selected for installation.

hold

A package marked to be on hold is kept on the same version, that is, no automatic new installs, upgrades or removals will be performed on them, unless these actions are requested explicitly,

option.

deinstall

The package is selected for deinstallation (i.e. we want to remove all files, except configuration files).

purge

The package is selected to be purged (i.e. we want to remove everything from system directories, even configuration files).

unknown

The package selection is unknown. A package that is also in a not-installed state, and with an ok flag will be forgotten in the next database store.

Package flags

ok A package marked ok is in a known state, but might need further processing.

reinstreq

A package marked reinstreq is broken and requires reinstallation. These packages cannot be removed, unless forced with option --force-remove-reinstreq.

-i, --install package-file...

Install the package. If --recursive or -R option is specified, package-file must refer to a directory instead.

Installation consists of the following steps:

1. **Extract the control files of the new package.**

2. **If another version of the same package was installed before the new installation, execute prerm script of the old package.**

3. **Run preinst script, if provided by the package.**

4. **Unpack the new files, and at the same time back up the old files, so that if something goes wrong, they can be restored.**

5. **If another version of the same package was installed before the new installation, execute the postrm script of the old package. Note that this script is executed after the preinst script of the new package, because new files are written at the same time old files are removed.**

6. **Configure the package. See --configure for detailed information about how this is done.**

--unpack package-file...

Unpack the package, but don't configure it. If --recursive or -R option is specified, package-file must refer to a directory instead.

Will process triggers for Pre-Depends unless --no-triggers has been specified.

--configure package...|-a|--pending

Configure a package which has been unpacked but not yet configured. If -a or --pending is given instead of package, all unpacked but unconfigured packages are configured.

To reconfigure a package which has already been configured, try the dpkg-reconfigure(8) command instead (which is part of the debconf project).

Configuring consists of the following steps:

1. **Unpack the conffiles, and at the same time back up the old conffiles, so that they can be restored if something goes wrong.**

2. **Run postinst script, if provided by the package.**

Will process triggers unless --no-triggers has been specified.

--triggers-only package...|-a|--pending

Processes only triggers (since dpkg 1.14.17). All pending triggers will be processed. If package names are supplied only those packages' triggers will be processed, exactly once each where necessary. Use of this option may leave packages in the improper triggers-awaited and triggers-pending states. This can be fixed later by running: dpkg --configure --pending.

-r, --remove package...|-a|--pending

Remove an installed package. This removes everything except conffiles and other data cleaned up by the postrm script, which may avoid having to reconfigure the package if it is reinstalled later (conffiles are configuration files that are listed in the DEBIAN/conffiles control file). If there is no DEBIAN/conffiles control file nor DEBIAN/postrm script, this command is equivalent to calling --purge. If -a or --pending is given instead of a package name, then all packages unpacked, but marked to be removed in file /var/lib/dpkg/status, are removed.

Removing of a package consists of the following steps:

1. Run prerm script.

2. Remove the installed files.

3. Run postrm script.

Will process triggers unless --no-triggers has been specified.

-P, --purge package...|-a|--pending

Purge an installed or already removed package. This removes everything, including conffiles, and anything else cleaned up from postrm. If -a or --pending is given instead of a package name, then all packages unpacked or removed, but marked to be purged in file /var/lib/dpkg/status, are purged.

Note: Some configuration files might be unknown to dpkg because they are created and handled separately through the configuration scripts. In that case, dpkg won't remove them by itself, but the package's postrm script (which is called by dpkg), has to take care of their removal during purge. Of course, this only applies to files in system directories, not configuration files written to individual users' home directories.

Purging of a package consists of the following steps:

1. Remove the package, if not already removed. See --remove for

2. Run postrm script.

Will process triggers unless --no-triggers has been specified.

-V, --verify [package-name...]

Verifies the integrity of package-name or all packages if omitted, by comparing information from the files installed by a package with the files metadata information stored in the dpkg database (since dpkg 1.17.2). The origin of the files metadata information in the database is the binary packages themselves. That metadata gets collected at package unpack time during the installation process.

Currently the only functional check performed is an md5sum verification of the file contents against the stored value in the files database. It will only get checked if the database contains the file md5sum. To check for any missing metadata in the database, the --audit command can be used. This is only an integrity check and should not be considered as any kind of security verification.

The output format is selectable with the --verify-format option, which by default uses the rpm format, but that might change in the future, and as such, programs parsing this command output should be

-C, --audit [package-name...]

Performs database sanity and consistency checks for package-name or all packages if omitted (per package checks since dpkg 1.17.10). For example, searches for packages that have been installed only partially on your system or that have missing, wrong or obsolete control data or files. dpkg will suggest what to do with them to get them fixed.

--update-avail [Packages-file]

--merge-avail [Packages-file]

Update dpkg's and dselect's idea of which packages are available. With action --merge-avail, old information is combined with information from Packages-file. With action --update-avail, old information is replaced with the information in the Packages-file. The Packages-file distributed with Debian is simply named ?Packages?. If the Packages-file argument is missing or named ?-? then it will be read from standard input (since dpkg 1.17.7). dpkg keeps its record of available packages in /var/lib/dpkg/available.

A simpler one-shot command to retrieve and update the available file is dselect update. Note that this file is mostly useless if you don't use dselect but an APT-based frontend: APT has its own system to keep track of available packages.

-A, --record-avail package-file...

Update dpkg and dselect's idea of which packages are available with information from the package package-file. If --recursive or -R option is specified, package-file must refer to a directory instead.

--forget-old-unavail

Now obsolete and a no-op as dpkg will automatically forget uninstalled unavailable packages (since dpkg 1.15.4), but only those that do not contain user information such as package selections.

--clear-avail

Erase the existing information about what packages are available.

--get-selections [package-name-pattern...]

Get list of package selections, and write it to stdout. Without a pattern, non-installed packages (i.e. those which have been previously purged) will not be shown.

--set-selections

Set package selections using file read from stdin. This file should be in the format ?package state?, where state is one of install, hold, deinstall or purge. Blank lines and comment lines

The available file needs to be up-to-date for this command to be useful, otherwise unknown packages will be ignored with a warning. See the --update-avail and --merge-avail commands for more information.

--clear-selections

Set the requested state of every non-essential package to deinstall (since dpkg 1.13.18). This is intended to be used immediately before --set-selections, to deinstall any packages not in list given to --set-selections.

--yet-to-unpack

Searches for packages selected for installation, but which for some reason still haven't been installed.

Note: This command makes use of both the available file and the package selections.

--predep-package

Print a single package which is the target of one or more relevant pre-dependencies and has itself no unsatisfied pre-dependencies.

If such a package is present, output it as a Packages file entry,

Note: This command makes use of both the available file and the package selections.

Returns 0 when a package is printed, 1 when no suitable package is available and 2 on error.

--add-architecture architecture

Add architecture to the list of architectures for which packages can be installed without using --force-architecture (since dpkg 1.16.2). The architecture dpkg is built for (i.e. the output of --print-architecture) is always part of that list.

--remove-architecture architecture

Remove architecture from the list of architectures for which packages can be installed without using --force-architecture (since dpkg 1.16.2). If the architecture is currently in use in the database then the operation will be refused, except if --force-architecture is specified. The architecture dpkg is built for (i.e. the output of --print-architecture) can never be removed from that list.

--print-architecture

Print architecture of packages dpkg installs (for example, ?i386?).

--print-foreign-architectures

Print a newline-separated list of the extra architectures dpkg is configured to allow packages to be installed for (since dpkg 1.16.2).

--assert-help

Give help about the --assert-feature options (since dpkg 1.21.0).

--assert-feature

Asserts that dpkg supports the requested feature. Returns 0 if the feature is fully supported, 1 if the feature is known but dpkg cannot provide support for it yet, and 2 if the feature is unknown. The current list of assertable features is:

support-predepends

Supports the Pre-Depends field (since dpkg 1.1.0).

working-epoch

Supports epochs in version strings (since dpkg 1.4.0.7).

long-filenames

Supports long filenames in deb(5) archives (since dpkg 1.4.1.17).

Supports multiple Conflicts and Replaces (since dpkg 1.4.1.19).

multi-arch

Supports multi-arch fields and semantics (since dpkg 1.16.2).

versioned-provides

Supports versioned Provides (since dpkg 1.17.11).

protected-field

Supports the Protected field (since dpkg 1.20.1).

--validate-thing string

Validate that the thing string has a correct syntax (since dpkg 1.18.16). Returns 0 if the string is valid, 1 if the string is invalid but might be accepted in lax contexts, and 2 if the string is invalid. The current list of validatable things is:

pkgname

Validates the given package name (since dpkg 1.18.16).

trigname

Validates the given trigger name (since dpkg 1.18.16).

archname

version

Validates the given version (since dpkg 1.18.16).

--compare-versions ver1 op ver2

Compare version numbers, where op is a binary operator. dpkg returns true (0) if the specified condition is satisfied, and false (1) otherwise. There are two groups of operators, which differ in how they treat an empty ver1 or ver2. These treat an empty version as earlier than any version: lt le eq ne ge gt. These treat an empty version as later than any version: lt-nl le-nl ge-nl gt-nl. These are provided only for compatibility with control file syntax: < << <= = >= >> >. The < and > operators are obsolete and should not be used, due to confusing semantics. To illustrate: 0.1 < 0.1 evaluates to true.

-?, --help

Display a brief help message.

--force-help

Give help about the --force-thing options.

-Dh, --debug=help

Give help about debugging options.

--version

Display dpkg version information.

When used with --robot, the output will be the program version number in a dotted numerical format, with no newline.

dpkg-deb actions

See dpkg-deb(1) for more information about the following actions, and other actions and options not exposed by the dpkg front-end.

-b, --build directory [archive|directory]

Build a deb package.

-c, --contents archive

List contents of a deb package.

-e, --control archive [directory]

Extract control-information from a package.

-x, --extract archive directory

Extract the files contained by package.

-X, --vextract archive directory

Extract and display the filenames contained by a package.

-f, --field archive [control-field...]

Display control field(s) of a package.

--ctrl-tarfile archive

Output the control tar-file contained in a Debian package.

--fsys-tarfile archive

Output the filesystem tar-file contained by a Debian package.

-I, --info archive [control-file...]

Show information about a package.

dpkg-query actions

See dpkg-query(1) for more information about the following actions,

and other actions and options not exposed by the dpkg front-end.

-l, --list package-name-pattern...

List packages matching given pattern.

-s, --status package-name...

Report status of specified package.

-L, --listfiles package-name...

List files installed to your system from package-name.

-S, --search filename-search-pattern...

Search for a filename from installed packages.

-p, --print-avail package-name...

Display details about package-name, as found in /var/lib/dpkg/available. Users of APT-based frontends should use apt show package-name instead.

OPTIONS

All options can be specified both on the command line and in the dpkg configuration file /etc/dpkg/dpkg.cfg or fragment files (with names matching this shell pattern '[0-9a-zA-Z_-]*') on the configuration directory /etc/dpkg/dpkg.cfg.d/. 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 ?#?).

--abort-after=number

Change after how many errors dpkg will abort. The default is 50.

-B, --auto-deconfigure

When a package is removed, there is a possibility that another installed package depended on the removed package. Specifying this option will cause automatic deconfiguration of the package which depended on the removed package.

-Doctal, --debug=octal

Switch debugging on. octal is formed by bitwise-ORing desired values together from the list below (note that these values may change in future releases). -Dh or --debug=help display these debugging values.

Number	Description
1	Generally helpful progress information
2	Invocation and status of maintainer scripts
10	Output for each file processed
100	Lots of output for each file processed
20	Output for each configuration file
200	Lots of output for each configuration file
40	Dependencies and conflicts
400	Lots of dependencies/conflicts output
10000	Trigger activation and processing
20000	Lots of output regarding triggers
40000	Silly amounts of output regarding triggers
1000	Lots of drivel about for example the dpkg/info dir
2000	Insane amounts of drivel

--force-things

--no-force-things, --refuse-things

Force or refuse (no-force and refuse mean the same thing) to do

below. --force-help displays a message describing them. Things marked with (*) are forced by default.

Warning: These options are mostly intended to be used by experts only. Using them without fully understanding their effects may break your whole system.

all:

Turns on (or off) all force options.

downgrade(*):

Install a package, even if newer version of it is already installed.

Warning: At present dpkg does not do any dependency checking on downgrades and therefore will not warn you if the downgrade breaks the dependency of some other package. This can have serious side effects, downgrading essential system components can even make your whole system unusable. Use with care.

configure-any:

Configure also any unpacked but unconfigured packages on which the current package depends.

Allow automatic installs, upgrades or removals of packages even when marked to be on ?hold?. Note: When these actions are requested explicitly, the ?hold? package selection state always gets ignored.

remove-reinstreq:

Remove a package, even if it's broken and marked to require reinstallation. This may, for example, cause parts of the package to remain on the system, which will then be forgotten by dpkg.

remove-protected:

Remove, even if the package is considered protected (since dpkg 1.20.1). Protected packages contain mostly important system boot infrastructure or are used for custom system-local meta-packages. Removing them might cause the whole system to be unable to boot or lose required functionality to operate, so use with caution.

remove-essential:

Remove, even if the package is considered essential. Essential packages contain mostly very basic Unix commands, required for the packaging system, for the operation of the system in general or during boot (although the latter should be converted to protected packages instead). Removing them might cause the whole

depends:

Turn all dependency problems into warnings. This affects the Pre-Depends and Depends fields.

depends-version:

Don't care about versions when checking dependencies. This affects the Pre-Depends and Depends fields.

breaks:

Install, even if this would break another package (since dpkg 1.14.6). This affects the Breaks field.

conflicts:

Install, even if it conflicts with another package. This is dangerous, for it will usually cause overwriting of some files. This affects the Conflicts field.

confmiss:

Always install the missing conffile without prompting. This is dangerous, since it means not preserving a change (removing) made to the file.

confnew:

did change, always install the new version without prompting, unless the --force-confdef is also specified, in which case the default action is preferred.

confold:

If a conffile has been modified and the version in the package did change, always keep the old version without prompting, unless the --force-confdef is also specified, in which case the default action is preferred.

confdef:

If a conffile has been modified and the version in the package did change, always choose the default action without prompting. If there is no default action it will stop to ask the user unless --force-confnew or --force-confold is also given, in which case it will use that to decide the final action.

confask:

If a conffile has been modified always offer to replace it with the version in the package, even if the version in the package did not change (since dpkg 1.15.8). If any of --force-confnew, --force-confold, or --force-confdef is also given, it will be used to decide the final action.

Overwrite one package's file with another's file.

overwrite-dir:

Overwrite one package's directory with another's file.

overwrite-diverted:

Overwrite a diverted file with an undiverted version.

statoverride-add:

Overwrite an existing stat override when adding it (since dpkg 1.19.5).

statoverride-remove:

Ignore a missing stat override when removing it (since dpkg 1.19.5).

security-mac(*):

Use platform-specific Mandatory Access Controls (MAC) based security when installing files into the filesystem (since dpkg 1.19.5). On Linux systems the implementation uses SELinux.

unsafe-io:

Do not perform safe I/O operations when unpacking (since dpkg 1.15.8.6). Currently this implies not performing file system

performance degradation on some file systems, unfortunately the ones that require the safe I/O on the first place due to their unreliable behaviour causing zero-length files on abrupt system crashes.

Note: For ext4, the main offender, consider using instead the mount option nodelalloc, which will fix both the performance degradation and the data safety issues, the latter by making the file system not produce zero-length files on abrupt system crashes with any software not doing syncs before atomic renames.

Warning: Using this option might improve performance at the cost of losing data, use with care.

script-chrootless:

Run maintainer scripts without chroot(2)ing into instdir even if the package does not support this mode of operation (since dpkg 1.18.5).

Warning: This can destroy your host system, use with extreme care.

architecture:

Process even packages with wrong or no architecture.

bad-version:

Process even packages with wrong versions (since dpkg 1.16.1).

bad-path:

PATH is missing important programs, so problems are likely.

not-root:

Try to (de)install things even when not root.

bad-verify:

Install a package even if it fails authenticity check.

--ignore-depends=package,...

Ignore dependency-checking for specified packages (actually, checking is performed, but only warnings about conflicts are given, nothing else). This affects the Pre-Depends, Depends and Breaks fields.

--no-act, --dry-run, --simulate

Do everything which is supposed to be done, but don't write any changes. This is used to see what would happen with the specified action, without actually modifying anything.

Be sure to give --no-act before the action-parameter, or you might

will first purge package ?foo? and then try to purge package ?--no-act?, even though you probably expected it to actually do nothing).

-R, --recursive

Recursively handle all regular files matching pattern *.deb found at specified directories and all of its subdirectories. This can be used with -i, -A, --install, --unpack and --record-avail actions.

-G Don't install a package if a newer version of the same package is already installed. This is an alias of --refuse-downgrade.

--admindir=dir

Set the administrative directory to directory. This directory contains many files that give information about status of installed or uninstalled packages, etc. Defaults to ?/var/lib/dpkg? if DPKG_ADMINDIR has not been set.

--instdir=dir

Set the installation directory, which refers to the directory where packages are to be installed. instdir is also the directory passed to chroot(2) before running package's installation scripts, which means that the scripts see instdir as a root directory. Defaults

--root=dir

Set the root directory to directory, which sets the installation directory to ?dir? and the administrative directory to ?dir/var/lib/dpkg?.

-O, --selected-only

Only process the packages that are selected for installation. The actual marking is done with dselect or by dpkg, when it handles packages. For example, when a package is removed, it will be marked selected for deinstallation.

-E, --skip-same-version

Don't install the package if the same version and architecture of the package is already installed.

Since dpkg 1.21.10, the architecture is also taken into account, which makes it possible to cross-grade packages or install additional co-installable instances with the same version, but different architecture.

--pre-invoke=command

--post-invoke=command

Set an invoke hook command to be run via ?sh -c? before or after

remove and purge actions (since dpkg 1.15.4), and add-architecture and remove-architecture actions (since dpkg 1.17.19). This option can be specified multiple times. The order the options are specified is preserved, with the ones from the configuration files taking precedence. The environment variable DPKG_HOOK_ACTION is set for the hooks to the current dpkg action.

Note: Front-ends might call dpkg several times per invocation, which might run the hooks more times than expected.

--path-exclude=glob-pattern

--path-include=glob-pattern

Set glob-pattern as a path filter, either by excluding or re-including previously excluded paths matching the specified patterns during install (since dpkg 1.15.8).

Warning: Take into account that depending on the excluded paths you might completely break your system, use with caution.

The glob patterns use the same wildcards used in the shell, were ?*? matches any sequence of characters, including the empty string and also ?/?. For example, ?/usr/*/READ*? matches ?/usr/share/doc/package/README?. As usual, ??? matches any single character (again, including ?/?). And ?[? starts a character

complementations. See glob(7) for detailed information about globbing. Note: The current implementation might re-include more directories and symlinks than needed, in particular when there is a more specific re-inclusion, to be on the safe side and avoid possible unpack failures; future work might fix this.

This can be used to remove all paths except some particular ones; a typical case is:

--path-exclude=/usr/share/doc/*
--path-include=/usr/share/doc/*/copyright

to remove all documentation files except the copyright files.

These two options can be specified multiple times, and interleaved with each other. Both are processed in the given order, with the last rule that matches a file name making the decision.

The filters are applied when unpacking the binary packages, and as such only have knowledge of the type of object currently being filtered (e.g. a normal file or a directory) and have not visibility of what objects will come next. Because these filters have side effects (in contrast to find(1) filters), excluding an exact pathname that happens to be a directory object like

pathname will be excluded (which could be automatically reincluded if the code sees the need). Any subsequent files contained within that directory will fail to unpack.

Hint: make sure the globs are not expanded by your shell.

--verify-format format-name

Sets the output format for the --verify command (since dpkg 1.17.2).

The only currently supported output format is rpm, which consists of a line for every path that failed any check. These lines have the following format:

missing [c] pathname [(error-message)]
??5?????? [c] pathname

The first 9 characters are used to report the checks result, either a literal missing when the file is not present or its metadata cannot be fetched, or one of the following special characters that report the result for each check:

??? Implies the check could not be done (lack of support, file permissions, etc).

?.? Implies the check passed.

?A-Za-z0-9?

Implies a specific check failed. The following positions and alphanumeric characters are currently supported:

1 ???

These checks are currently not supported, will always be ???.

2 ?M?

The file mode check failed (since dpkg 1.21.0). Because pathname metadata is currently not tracked, this check can only be partially emulated via a very simple heuristic for pathnames that have a known digest, which implies they should be regular files, where the check will fail if the pathname is not a regular file on the filesystem. This check will currently never succeed as it does not have enough information available.

3 ?5?

The digest check failed, which means the file contents have changed. This is only an integrity check and should not be considered as any kind of security verification.

4-9 ???

These checks are currently not supported, will always be ???.

The line is followed by a space and an attribute character. The following attribute character is supported:

?c? The pathname is a conffile.

Finally followed by another space and the pathname.

In case the entry was of the missing type, and the file was not actually present on the filesystem, then the line is followed by a space and the error message enclosed within parenthesis.

--status-fd n

Send machine-readable package status and progress information to file descriptor n. This option can be specified multiple times. The information is generally one record per line, in one of the following forms:

status: package: status

Package status changed; status is as in the status file.

An error occurred. Any possible newlines in extended-error-message will be converted to spaces before output.

status: file : conffile-prompt : 'real-old' 'real-new' useredited distedited

User is being asked a conffile question.

processing: stage: package

Sent just before a processing stage starts. stage is one of upgrade, install (both sent before unpacking), configure, trigproc, disappear, remove, purge.

--status-logger=command

Send machine-readable package status and progress information to the shell command's standard input, to be run via ?sh -c? (since dpkg 1.16.0). This option can be specified multiple times. The output format used is the same as in --status-fd.

--log=filename

Log status change updates and actions to filename, instead of the default /var/log/dpkg.log. If this option is given multiple times, the last filename is used. Log messages are of the form:

YYYY-MM-DD HH:MM:SS startup type command

of unpack or install) or packages (with a command of configure, triggers-only, remove or purge).

YYYY-MM-DD HH:MM:SS status state pkg installed-version
For status change updates.

YYYY-MM-DD HH:MM:SS action pkg installed-version available-version
For actions where action is one of install, upgrade, configure, trigproc, disappear, remove or purge.

YYYY-MM-DD HH:MM:SS conffile filename decision
For conffile changes where decision is either install or keep.

--robot
Use a machine-readable output format. This provides an interface for programs that need to parse the output of some of the commands that do not otherwise emit a machine-readable output format. No localization will be used, and the output will be modified to make it easier to parse.

The only currently supported command is --version.

--no-pager
Disables the use of any pager when showing information (since dpkg

--no-debsig

Do not try to verify package signatures.

--no-triggers

Do not run any triggers in this run (since dpkg 1.14.17), but activations will still be recorded. If used with --configure package or --triggers-only package then the named package postinst will still be run even if only a triggers run is needed. Use of this option may leave packages in the improper triggers-awaited and triggers-pending states. This can be fixed later by running: dpkg --configure --pending.

--triggers

Cancels a previous --no-triggers (since dpkg 1.14.17).

EXIT STATUS

0 The requested action was successfully performed. Or a check or assertion command returned true.

1 A check or assertion command returned false.

2 Fatal or unrecoverable error due to invalid command-line usage, or interactions with the system, such as accesses to the database,

ENVIRONMENT

External environment

PATH

This variable is expected to be defined in the environment and point to the system paths where several required programs are to be found. If it's not set or the programs are not found, dpkg will abort.

HOME

If set, dpkg will use it as the directory from which to read the user specific configuration file.

TMPDIR

If set, dpkg will use it as the directory in which to create temporary files and directories.

SHELL

The program dpkg will execute when starting a new interactive shell, or when spawning a command via a shell.

PAGER

DPKG_PAGER

The program dpkg will execute when running a pager, which will be

differences. If SHELL is not set, ?sh? will be used instead. The DPKG_PAGER overrides the PAGER environment variable (since dpkg 1.19.2).

DPKG_COLORS

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

DPKG_DEBUG

Sets the debug mask (since dpkg 1.21.10) from an octal value. The currently accepted flags are described in the --debug option.

DPKG_FORCE

Sets the force flags (since dpkg 1.19.5). When this variable is present, no built-in force defaults will be applied. If the variable is present but empty, all force flags will be disabled.

DPKG_ADMINDIR

If set and the --admindir or --root options have not been specified, it will be used as the dpkg administrative directory (since dpkg 1.20.0).

DPKG_FRONTEND_LOCKED

Set by a package manager frontend to notify dpkg that it should not

Internal environment

LESS

Defined by dpkg to ?-FRSXMQ?, if not already set, when spawning a pager (since dpkg 1.19.2). To change the default behavior, this variable can be preset to some other value including an empty string, or the PAGER or DPKG_PAGER variables can be set to disable specific options with ?-+?, for example DPKG_PAGER="less -+F".

DPKG_ROOT

Defined by dpkg on the maintainer script environment to indicate which installation to act on (since dpkg 1.18.5). The value is intended to be prepended to any path maintainer scripts operate on. During normal operation, this variable is empty. When installing packages into a different instdir, dpkg normally invokes maintainer scripts using chroot(2) and leaves this variable empty, but if --force-script-chrootless is specified then the chroot(2) call is skipped and instdir is non-empty.

DPKG_ADMINDIR

Defined by dpkg on the maintainer script environment to indicate the dpkg administrative directory to use (since dpkg 1.16.0). This variable is always set to the current --admindir value.

Defined by dpkg on the subprocesses environment to all the currently enabled force option names separated by commas (since dpkg 1.19.5).

DPKG_SHELL_REASON

Defined by dpkg on the shell spawned on the conffile prompt to examine the situation (since dpkg 1.15.6). Current valid value: conffile-prompt.

DPKG_CONFFILE_OLD

Defined by dpkg on the shell spawned on the conffile prompt to examine the situation (since dpkg 1.15.6). Contains the path to the old conffile.

DPKG_CONFFILE_NEW

Defined by dpkg on the shell spawned on the conffile prompt to examine the situation (since dpkg 1.15.6). Contains the path to the new conffile.

DPKG_HOOK_ACTION

Defined by dpkg on the shell spawned when executing a hook action (since dpkg 1.15.4). Contains the current dpkg action.

DPKG_RUNNING_VERSION

of the currently running dpkg instance (since dpkg 1.14.17).

DPKG_MAINTSCRIPT_PACKAGE

Defined by dpkg on the maintainer script environment to the (non-arch-qualified) package name being handled (since dpkg 1.14.17).

DPKG_MAINTSCRIPT_PACKAGE_REFCOUNT

Defined by dpkg on the maintainer script environment to the package reference count, i.e. the number of package instances with a state greater than not-installed (since dpkg 1.17.2).

DPKG_MAINTSCRIPT_ARCH

Defined by dpkg on the maintainer script environment to the architecture the package got built for (since dpkg 1.15.4).

DPKG_MAINTSCRIPT_NAME

Defined by dpkg on the maintainer script environment to the name of the script running, one of preinst, postinst, prerm or postrm (since dpkg 1.15.7).

DPKG_MAINTSCRIPT_DEBUG

Defined by dpkg on the maintainer script environment to a value (?0? or ?1?) noting whether debugging has been requested (with the --debug option) for the maintainer scripts (since dpkg 1.18.4).

FILES

/etc/dpkg/dpkg.cfg.d/[0-9a-zA-Z_-]*

Configuration fragment files (since dpkg 1.15.4).

/etc/dpkg/dpkg.cfg

Configuration file with default options.

/var/log/dpkg.log

Default log file (see /etc/dpkg/dpkg.cfg and option --log).

The other files listed below are in their default directories, see option --admindir to see how to change locations of these files.

/var/lib/dpkg/available

List of available packages.

/var/lib/dpkg/status

Statuses of available packages. This file contains information about whether a package is marked for removing or not, whether it is installed or not, etc. See section "INFORMATION ABOUT PACKAGES" for more info.

The status file is backed up daily in /var/backups. It can be useful if it's lost or corrupted due to filesystems troubles.

The format and contents of a binary package are described in deb(5).

Filesystem filenames

During unpacking and configuration dpkg uses various filenames for backup and rollback purposes. The following is a simplified explanation of how these filenames get used during package installation.

*.dpkg-new

During unpack, dpkg extracts new filesystem objects into pathname.dpkg-new (except for existing directories or symlinks to directories which get skipped), once that is done and after having performed backups of the old objects, the objects get renamed to pathname.

*.dpkg-tmp

During unpack, dpkg makes backups of the old filesystem objects into pathname.dpkg-tmp after extracting the new objects. These backups are performed as either a rename for directories (but only if they switch file type), a new symlink copy for symlinks, or a hard link for any other filesystem object, except for conffiles which get no backups because they are processed at a later stage.

In case of needing to rollback, these backups get used to restore

automatically after the installation is complete.

*.dpkg-old

During configuration, when installing a new version, dpkg can make a backup of the previous modified conffile into pathname.dpkg-old.

*.dpkg-dist

During configuration, when keeping the old version, dpkg can make a backup of the new unmodified conffile into pathname.dpkg-dist.

SECURITY

Any operation that needs write access to the database or the filesystem is considered a privileged operation that might allow root escalation. These operations must never be delegated to an untrusted user or be done on untrusted packages, as that might allow root access to the system.

Some operations (such as package verification) might need root privileges to be able to access files on the filesystem that would otherwise be inaccessible due to restricted permissions, but should otherwise work normally and produce appropriate messages in those cases.

Query operations should never require root, and delegating their

security implications (such as privilege escalation), for example when a pager is automatically invoked by the tool.

See also the SECURITY section of the dpkg-deb(1) and dpkg-split(1) manual pages.

BUGS

--no-act usually gives less information than might be helpful.

EXAMPLES

To list installed packages related to the editor vi(1) (note that dpkg-query does not load the available file anymore by default, and the dpkg-query --load-avail option should be used instead for that):

```
dpkg -l '*vi*'
```

To see the entries in /var/lib/dpkg/available of two packages:

```
dpkg --print-avail vim neovim | less
```

To search the listing of packages yourself:

```
dpkg --print-avail | less
```

dpkg -r neovim

To install a package, you first need to find it in an archive or media disc. When using an archive based on a pool structure, knowing the archive area and the name of the package is enough to infer the pathname:

dpkg -i /media/bdrom/pool/main/v/vim/vim_9.0.2018-1_amd64.deb

To make a local copy of the package selection states:

dpkg --get-selections >myselections

You might transfer this file to another computer, and after having updated the available file there with your package manager frontend of choice (see <https://wiki.debian.org/Teams/Dpkg/FAQ#set-selections> for more details), for example:

apt-cache dumpavail | dpkg --merge-avail

you can install it with:

dpkg --clear-selections

Note that this will not actually install or remove anything, but just set the selection state on the requested packages. You will need some other application to actually download and install the requested packages. For example, run apt-get dselect-upgrade.

Ordinarily, you will find that dselect(1) provides a more convenient way to modify the package selection states.

ADDITIONAL FUNCTIONALITY

Additional functionality can be gained by installing any of the following packages: apt, aptitude and debsig-verify.

SEE ALSO

aptitude(8), apt(8), dselect(1), dpkg-deb(1), dpkg-query(1), deb(5), deb-control(5), dpkg.cfg(5), and dpkg-reconfigure(8).

AUTHORS

See /usr/share/doc/dpkg/THANKS for the list of people who have contributed to dpkg.