



## ***Rocky Enterprise Linux 9.2 Manual Pages on command 'grep-status.1'***

**C:\>man grep-status.1**

grep-DCTRL(1)                    Debian user's manual                    grep-DCTRL(1)

### NAME

grep-dctrl, grep-status, grep-available, grep-aptavail, grep-debtags - grep Debian control files

### SYNOPSIS

command --copying|-C | --help|-h | --version|-V

command [options] filter [ file... ]

where command is one of grep-dctrl, grep-status, grep-available, grep-aptavail and grep-debtags.

### DESCRIPTION

The grep-dctrl program can answer such questions as What is the Debian package foo?, Which version of the Debian package bar is now current?, Which Debian packages does John Doe maintain?, Which Debian packages are somehow related to the Scheme programming language?, and with some help, Who maintain the essential packages of a Debian system?, given a useful input file.

The programs grep-available, grep-status, grep-aptavail and grep-debtags are aliases of (actually, symbolic links to) grep-dctrl. These aliases use as their default input the dpkg(1) available and status files, the apt-cache dumpavail output and the debtags dumpavail output, respectively.

grep-dctrl is a specialised grep program that is meant for processing any file which has the general format of a Debian package control file, as described in the Debian Policy. These include the dpkg available file, the dpkg status file, and

the Packages files on a distribution medium (such as a Debian CD-ROM or an FTP site carrying Debian).

You must give a filter expression on the command line. The filter defines which kind of paragraphs (aka package records) are output. A simple filter is a search pattern along with any options that modify it. Possible modifiers are `--eregex`, `--field`, `--ignore-case`, `--regex` and `--exact-match`, along with their single-letter equivalents. By default, the search is a case-sensitive fixed substring match on each paragraph (in other words, package record) in the input. With suitable modifiers, this can be changed: the search can be case-insensitive and the pattern can be seen as an extended POSIX regular expression.

Filters can be combined to form more complex filters using the connectives `--and`, `--or` and `--not`. Parentheses (which usually need to be escaped for the shell) can be used for grouping.

By default, the full matching paragraphs are printed on the standard output; specific fields can be selected for output with the `-s` option.

After the filter expression comes zero or more file names. The file name `-` is taken to mean the standard input stream. The files are searched in order but separately; they are not concatenated together. In other words, the end of a file always implies the end of the current paragraph.

If no file names are specified, the program name is used to identify a default input file. The program names are matched with the base form of the name of the current program (the 0'th command line argument, if you will).

## OPTIONS

### Specifying the search pattern

`--pattern=pattern`

Specify a pattern to be searched. This switch is not generally needed, as the pattern can be given by itself. However, patterns that start with a dash (`-`) must be given using this switch, so that they wouldn't be mistaken for switches.

### Modifiers of simple filters

`-F field,field, ... | --field=field,field, ...`

Restrict pattern matching to the fields given. Multiple field names in one `-F` option and multiple `-F` options in one simple filter are allowed. The

search named by the filter will be performed among all the fields named, and as soon as any one of them matches, the whole simple filter is considered matching.

A field specification can contain a colon (:). In such a case, the part up to the colon is taken as the name of the field to be searched in, and the part after the colon is taken as the name of the field whose content is to be used if the field to search in is empty.

**-P** Shorthand for **-FPackage**.

**-S** Shorthand for **-FSource:Package**.

**-e, --eregex**

Regard the pattern of the current simple filter as an extended POSIX regular expression

**-r, --regex**

Regard the pattern of the current simple filter as a standard POSIX regular expression.

**-i, --ignore-case**

Ignore case when looking for a match in the current simple filter.

**-X, --exact-match**

Do an exact match (as opposed to a substring match) in the current simple filter.

**-w, --whole-pkg**

Do an extended regular expression match on whole package names, assuming the syntax of inter-package relationship fields such as Depends, Recommends, ...

When this flag is given you should not worry about sub-package names such as "libpcre3" also matching "libpcre3-dev". This flag implies (and is incompatible with) **-e**.

**--eq** Do an equality comparison under the Debian version number system. If the pattern or the field to be searched in is not a valid Debian version number, the paragraph is regarded as not matching. As a special case, this is capable of comparing simple nonnegative integers for equality.

**--lt** Do a strictly-less-than comparison under the Debian version number system. If the pattern or the field to be searched in is not a valid Debian version number, the paragraph is regarded as not matching. As a special case, this

is capable of comparing simple nonnegative integers.

--le Do an less-than-or-equal comparison under the Debian version number system.

If the pattern or the field to be searched in is not a valid Debian version number, the paragraph is regarded as not matching. As a special case, this is capable of comparing simple nonnegative integers.

--gt Do an strictly-greater-than comparison under the Debian version number system.

If the pattern or the field to be searched in is not a valid Debian version number, the paragraph is regarded as not matching. As a special case, this is capable of comparing simple nonnegative integers.

--ge Do an greater-than-or-equal comparison under the Debian version number system.

If the pattern or the field to be searched in is not a valid Debian version number, the paragraph is regarded as not matching. As a special case, this is capable of comparing simple nonnegative integers.

#### Combining filters

!, --not, !

Match if the following filter does not match.

-o, --or

Match if either one or both of the preceding and following filters matches.

-a, --and

Match if both the preceding and the following filter match.

( ... )

Parentheses can be used for grouping. Note that they need to be escaped for most shells. Filter modifiers can be given before the opening parentheses; they will be treated as if they had been repeated for each simple filter inside the parentheses.

#### Output format modifiers

-l, --files-with-matches

Output only the file names, each on its own line, of those files that contain at least one matching paragraph. This is incompatible with the -v and -L options, and all other output format modifiers will be ignored.

-L, --files-without-matches

Output only the file names, each on its own line, of those files that do not contain any matching paragraphs. This is incompatible with the -v and -l

options, and all other output format modifiers will be ignored.

`-s field,field, ... | --show-field=field,field, ...`

Show only the body of these fields from the matching paragraphs. The field names must not include any colons or commas. Commas are used to delimit field names in the argument to this option. The fields are shown in the order given here. See also the option `-l`. Note that in the absence of the `--ensure-dctrl` option, if only one field is selected, no paragraph separator is output.

`-l, --invert-show`

Invert the meaning of option `-s`: show only the fields that have not been named using a `-s` option. As an artefact of the implementation, the order of the fields in the original paragraph is not preserved.

A field specification can contain a colon. In such a case, the part up to the colon is taken as the name of the field to be shown, and the part after the colon is taken as the name of the field whose content is to be used if the field to be shown is empty.

`-d` Show only the first line of the Description field from the matching paragraphs. If no `-s` option is specified, this option also effects `-s Description`; if there is a `-s` option but it does not include the Description field name, one is appended to the option. Thus the Description field's location in the output is determined by the `-s` option, if any, the last field being the default.

`-n, --no-field-names`

Suppress field names when showing specified fields, only their bodies are shown. Each field is printed in its original form without the field name, the colon after it and any whitespace preceding the start of the body.

`-v, --invert-match`

Instead of showing all the paragraphs that match, show those paragraphs that do not match.

`-c, --count`

Instead of showing the paragraphs that match (or, with `-v`, that don't match), show the count of those paragraphs.

`-q, --quiet, --silent`

Output nothing to the standard output stream. Instead, exit immediately after finding the first match.

## Miscellaneous

### --ensure-dctrl

Ensure that the output is in dctrl format, specifically that there always is an empty line separating paragraphs. This option is not honored if the -n option has been selected, as that option deliberately requests a non-dctrl format for the output. In a future version, this option may be made the default behaviour.

### --compat

Override any --ensure-dctrl option given earlier on the command line.

### --ignore-parse-errors

Ignore errors in parsing input. A paragraph which cannot be parsed is ignored in its entirety, and the next paragraph is assumed to start after the first newline since the location of the error.

### --debug-optparse

Show how the current command line has been parsed.

### --errorlevel=level

Set log level to level. level is one of fatal, important, informational and debug, but the last may not be available, depending on the compile-time options. These categories are given here in order; every message that is emitted when fatal is in effect, will be emitted in the important error level, and so on. The default is important.

### -V, --version

Print out version information.

### -C, --copying

Print out the copyright license. This produces much output; be sure to redirect or pipe it somewhere (such as your favourite pager).

### -h, --help

Print out a help summary.

## EXAMPLES

The almost simplest use of this program is to print out the status or available record of a package. In this respect, grep-dctrl is like dpkg -s or dpkg

--print-avail. To print out the status record of the package "mixal", do

```
% grep-status -PX mixal
```

and to get its available record, use

```
% grep-available -PX mixal
```

In fact, you can ask for the record of the "mixal" package from any Debian control file. Say, you have the Debian 6.0 CD-ROM's Packages file in the current directory; now you can do a

```
% grep-dctrl -PX mixal Packages
```

But `grep-dctrl` can do more than just emulate `dpkg`. It can more-or-less emulate `apt-cache`! That program has a search feature that searches package descriptions.

But we can do that too:

```
% grep-available -F Description foo
```

searches for the string "foo" case-sensitively in the descriptions of all available packages. If you want case-insensitivity, use

```
% grep-available -F Description -i foo
```

Truth to be told, `apt-cache` searches package names, too. We can separately search in the names; to do so, do

```
% grep-available -F Package foo
```

or

```
% grep-available -P foo
```

which is pretty much the same thing. We can also search in both descriptions and names; if match is found in either, the package record is printed:

```
% grep-available -P -F Description foo
```

or

```
% grep-available -F Package -F Description foo
```

This kind of search is the exactly same that `apt-cache` does.

Here's one thing neither `dpkg` nor `apt-cache` do. Search for a string in the whole `status` or `available` file (or any Debian control file, for that matter) and print out all package records where we have a match. Try

```
% grep-available dpkg
```

sometime and watch how thoroughly `dpkg` has infiltrated Debian.

All the above queries were based on simple substring searches. But `grep-dctrl` can handle regular expressions in the search pattern. For example, to see the status

records of all packages with either "apt" or "dpkg" in their names, use

```
% grep-status -P -e 'apt|dpkg'
```

Now that we have seen all these fine and dandy queries, you might begin to wonder whether it is necessary to always see the whole paragraph. You may be, for exam-

ple, interested only in the dependency information of the packages involved. Fine.

To show the depends lines of all packages maintained by me, do a

```
% grep-available -F Maintainer -s Depends 'ajk@debian.org'
```

If you want to see the packages' names, too, use

```
% grep-available -F Maintainer -s Package,Depends \  
'ajk@debian.org'
```

Note that there must be no spaces in the argument to the -s switch.

More complex queries are also possible. For example, to see the list of packages maintained by me and depending on libc6, do

```
% grep-available -F Maintainer 'ajk@debian.org' \  
-a -F Depends libc6 -s Package,Depends
```

Remember that you can use other UNIX filters to help you, too. Ever wondered, who's the most active Debian developer based on the number of source packages being maintained? Easy. You just need to have a copy of the most recent Sources file from any Debian mirror.

```
% grep-dctrl -n -s Maintainer " Sources | sort | \  
uniq -c | sort -nr
```

This example shows a neat trick: if you want to selectively show only some field of all packages, just supply an empty pattern.

The term "bogopackage" means the count of the packages that a Debian developer maintains. To get the bogopackage count for the maintainer of dctrl-tools, say

```
% grep-available -c -FMaintainer \  
" `grep-available -sMaintainer -n -PX dctrl-tools`"
```

Sometimes it is useful to output the data of several fields on the same line. For example, the following command outputs the list of installed packages, sorted by their Installed-Size.

```
% grep-status -FStatus -sInstalled-Size,Package -n \  
"install ok installed" -a -FInstalled-Size --gt 0 \  
| paste -sd " \n" | sort -n
```

Note that there should be exactly 2 spaces in the " \n" string.

Another usual use-case is looking for packages that have another one as build dependency:

```
% grep-dctrl -s Package -F Build-Depends,Build-Depends-Indep \  
quilt /var/lib/apt/lists/*Sources
```

These examples cover a lot of typical uses of this utility, but not all possible uses. Use your imagination! The building blocks are there, and if something's missing, let me know.

## DIAGNOSTICS

In the absence of errors, the exit code 0 is used if at least one match was found, and the exit code 1 is used if no matches were found. If there were errors, the exit code is 2, with one exception. If the `-q`, `--quiet` or `--silent` options are used, the exit code 0 is used when a match is found regardless of whether there have been non-fatal errors.

These messages are emitted in log levels fatal and important. Additional messages may be provided by the system libraries. This list is incomplete.

A pattern is mandatory

You must specify a pattern to be searched for.

malformed filter

No filter was specified, but one is required.

cannot find enough memory

More memory was needed than was available. This error may be transient, that is, if you try again, all may go well.

cannot suppress field names when showing whole paragraphs

When you do not use the `-s` switch, `grep-dctrl` just passes the matching paragraphs through, not touching them any way. This means, for example, that you can only use `-n` when you use `-s`.

inconsistent modifiers of simple filters

Conflicting modifiers of simple filters were used; for example, perhaps both `-X` and `-e` were specified for the same simple filter.

missing ')' in command line

There were more opening than closing parentheses in the given filter.

no such log level

The argument to --errorlevel was invalid.

too many file names

The number of file names specified in the command line exceeded a compile-time limit.

too many output fields

The argument to -s had too many field names in it. This number is limited to 256.

unexpected ')' in command line

There was no opening parenthesis that would match some closing parenthesis in the command line.

## FILES

/var/lib/dpkg/available

The default input file of grep-available.

/var/lib/dpkg/status

The default input file of grep-status.

## AUTHOR

The program and this manual page were written by Antti-Juhani Kaijanaho <gaia@iki.fi>. Bill Allombert <ballombe@debian.org> provided one of the examples in the manual page.

## SEE ALSO

Debian Policy Manual. Published as the Debian package debian-policy. Also available in the Debian website.

apt-cache(1), ara(1), dpkg-awk(1), sgrep(1), dpkg(8)

Debian Project

2013-11-26

GREP-DCTRL(1)