NAME

grep, egrep, fgrep, rgrep − print lines matching a pattern

SYNOPSIS

grep [*OPTIONS*] *PATTERN* [*FILE*...]
grep [*OPTIONS*] [**−e** *PATTERN* | **−f** *FILE*] [*FILE*...]

DESCRIPTION

grep searches the named input *FILE*s (or standard input if no files are named, or if a single hyphen-minus (−) is given as file name) for lines containing a match to the given *PATTERN*. By default, **grep** prints the matching lines.

In addition, three variant programs **egrep, fgrep** and **rgrep** are available. **egrep** is the same as **grep −E**. **fgrep** is the same as **grep −F**. **rgrep** is the same as **grep −r**. Direct invocation as either **egrep** or **fgrep** is deprecated, but is provided to allow historical applications that rely on them to run unmodified.

OPTIONS

Generic Program Information

−−help Print a usage message briefly summarizing these command-line options and the bug-reporting address, then exit.

−V, **−−version**
Print the version number of **grep** to the standard output stream. This version number should be included in all bug reports (see below).

Matcher Selection

−E, **−−extended−regexp**
Interpret *PATTERN* as an extended regular expression (ERE, see below). (**−E** is specified by POSIX.)

−F, **−−fixed−strings**
Interpret *PATTERN* as a list of fixed strings, separated by newlines, any of which is to be matched. (**−F** is specified by POSIX.)

−G, **−−basic−regexp**
Interpret *PATTERN* as a basic regular expression (BRE, see below). This is the default.

−P, **−−perl−regexp**
Interpret *PATTERN* as a Perl regular expression (PCRE, see below). This is highly experimental and **grep −P** may warn of unimplemented features.

Matching Control

−e *PATTERN*, **−−regexp=***PATTERN*
Use *PATTERN* as the pattern. This can be used to specify multiple search patterns, or to protect a pattern beginning with a hyphen (−). (**−e** is specified by POSIX.)

−f *FILE*, **−−file=***FILE*
Obtain patterns from *FILE*, one per line. The empty file contains zero patterns, and therefore matches nothing. (**−f** is specified by POSIX.)

−i, **−−ignore−case**
Ignore case distinctions in both the *PATTERN* and the input files. (**−i** is specified by POSIX.)

−v, **−−invert−match**
Invert the sense of matching, to select non-matching lines. (**−v** is specified by POSIX.)

−w, **−−word−regexp**
Select only those lines containing matches that form whole words. The test is that the matching substring must either be at the beginning of the line, or preceded by a non-word constituent character. Similarly, it must be either at the end of the line or followed by a non-word constituent character. Word-constituent characters are letters, digits, and the underscore.

−x, **−−line−regexp**
Select only those matches that exactly match the whole line. (**−x** is specified by POSIX.)

−y Obsolete synonym for **−i**.

General Output Control

−c, **−−count**
Suppress normal output; instead print a count of matching lines for each input file. With the **−v**, **−−invert−match** option (see below), count non-matching lines. (**−c** is specified by POSIX.)

−−color[=*WHEN*], **−−colour**[=*WHEN*]
Surround the matched (non-empty) strings, matching lines, context lines, file names, line numbers, byte offsets, and separators (for fields and groups of context lines) with escape sequences to display them in color on the terminal. The colors are defined by the environment variable **GREP_COLORS**. The deprecated environment variable **GREP_COLOR** is still supported, but its setting does not have priority. *WHEN* is **never**, **always**, or **auto**.

−L, **−−files−without−match**
Suppress normal output; instead print the name of each input file from which no output would normally have been printed. The scanning will stop on the first match.

−l, **−−files−with−matches**
Suppress normal output; instead print the name of each input file from which output would normally have been printed. The scanning will stop on the first match. (**−l** is specified by POSIX.)

−m *NUM*, **−−max−count=***NUM*
Stop reading a file after *NUM* matching lines. If the input is standard input from a regular file, and *NUM* matching lines are output, **grep** ensures that the standard input is positioned to just after the last matching line before exiting, regardless of the presence of trailing context lines. This enables a calling process to resume a search. When **grep** stops after *NUM* matching lines, it outputs any trailing context lines. When the **−c** or **−−count** option is also used, **grep** does not output a count greater than *NUM*. When the **−v** or **−−invert−match** option is also used, **grep** stops after outputting *NUM* non-matching lines.

−o, **−−only−matching**
Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.

−q, **−−quiet**, **−−silent**
Quiet; do not write anything to standard output. Exit immediately with zero status if any match is found, even if an error was detected. Also see the **−s** or **−−no−messages** option. (**−q** is specified by POSIX.)

−s, **−−no−messages**
Suppress error messages about nonexistent or unreadable files. Portability note: unlike GNU **grep**, 7th Edition Unix **grep** did not conform to POSIX, because it lacked **−q** and its **−s** option behaved like GNU **grep**'s **−q** option. USG-style **grep** also lacked **−q** but its **−s** option behaved like GNU **grep**. Portable shell scripts should avoid both **−q** and **−s** and should redirect standard and error output to **/dev/null** instead. (**−s** is specified by POSIX.)

Output Line Prefix Control

−b, **−−byte−offset**
Print the 0-based byte offset within the input file before each line of output. If **−o** (**−−only−matching**) is specified, print the offset of the matching part itself.

−H, **−−with−filename**
Print the file name for each match. This is the default when there is more than one file to search.

−h, **−−no−filename**
Suppress the prefixing of file names on output. This is the default when there is only one file (or only standard input) to search.

−−label=*LABEL*
Display input actually coming from standard input as input coming from file *LABEL*. This is especially useful when implementing tools like **zgrep**, e.g., **gzip -cd foo.gz | grep --label=foo -H something**. See also the **−H** option.

−n, **−−line−number**
Prefix each line of output with the 1-based line number within its input file. (**−n** is specified by POSIX.)

−T, **−−initial−tab**
Make sure that the first character of actual line content lies on a tab stop, so that the alignment of tabs looks normal. This is useful with options that prefix their output to the actual content: **−H**,**−n**, and **−b**. In order to improve the probability that lines from a single file will all start at the same column, this also causes the line number and byte offset (if present) to be printed in a minimum size field width.

−u, **−−unix−byte−offsets**
Report Unix-style byte offsets. This switch causes **grep** to report byte offsets as if the file were a Unix-style text file, i.e., with CR characters stripped off. This will produce results identical to running **grep** on a Unix machine. This option has no effect unless **−b** option is also used; it has no effect on platforms other than MS-DOS and MS-Windows.

−Z, **−−null**
Output a zero byte (the ASCII **NUL** character) instead of the character that normally follows a file name. For example, **grep −lZ** outputs a zero byte after each file name instead of the usual newline. This option makes the output unambiguous, even in the presence of file names containing unusual characters like newlines. This option can be used with commands like **find −print0**, **perl −0**, **sort −z**, and **xargs −0** to process arbitrary file names, even those that contain newline characters.

Context Line Control

−A *NUM*, **−−after−context=***NUM*
Print *NUM* lines of trailing context after matching lines. Places a line containing a group separator (**−−**) between contiguous groups of matches. With the **−o** or **−−only−matching** option, this has no effect and a warning is given.

−B *NUM*, **−−before−context=***NUM*
Print *NUM* lines of leading context before matching lines. Places a line containing a group separator (**−−**) between contiguous groups of matches. With the **−o** or **−−only−matching** option, this has no effect and a warning is given.

−C *NUM*, *−NUM*, **−−context=***NUM*
Print *NUM* lines of output context. Places a line containing a group separator (**−−**) between contiguous groups of matches. With the **−o** or **−−only−matching** option, this has no effect and a warning is given.

File and Directory Selection

−a, **−−text**
Process a binary file as if it were text; this is equivalent to the **−−binary−files=text** option.

−−binary−files=*TYPE*
If the first few bytes of a file indicate that the file contains binary data, assume that the file is of type *TYPE*. By default, *TYPE* is **binary**, and **grep** normally outputs either a one-line message saying that a binary file matches, or no message if there is no match. If *TYPE* is **without-match**, **grep** assumes that a binary file does not match; this is equivalent to the **−I** option. If *TYPE* is **text**, **grep** processes a binary file as if it were text; this is equivalent to the **−a** option. *Warning:* **grep −−binary−files=text** might output binary garbage, which can have nasty side effects if the output is a terminal and if the terminal driver interprets some of it as commands.

−D *ACTION*, **−−devices=***ACTION*
If an input file is a device, FIFO or socket, use *ACTION* to process it. By default, *ACTION* is **read**, which means that devices are read just as if they were ordinary files. If *ACTION* is **skip**, devices are silently skipped.

−d *ACTION*, **−−directories=***ACTION*
If an input file is a directory, use *ACTION* to process it. By default, *ACTION* is **read**, i.e., read directories just as if they were ordinary files. If *ACTION* is **skip**, silently skip directories. If *ACTION* is **recurse**, read all files under each directory, recursively, following symbolic links only if they are on the command line. This is equivalent to the **−r** option.

−−exclude=*GLOB*
Skip files whose base name matches *GLOB* (using wildcard matching). A file-name glob can use *****, **?**, and **[...]** as wildcards, and \ to quote a wildcard or backslash character literally.

−−exclude-from=*FILE*
Skip files whose base name matches any of the file-name globs read from *FILE* (using wildcard matching as described under **−−exclude**).

−−exclude-dir=*DIR*
Exclude directories matching the pattern *DIR* from recursive searches.

−I Process a binary file as if it did not contain matching data; this is equivalent to the **−−binary−files=without-match** option.

−−include=*GLOB*
Search only files whose base name matches *GLOB* (using wildcard matching as described under **−−exclude**).

−r, **−−recursive**
Read all files under each directory, recursively, following symbolic links only if they are on the command line. This is equivalent to the **−d recurse** option.

−R, **−−dereference−recursive**
Read all files under each directory, recursively. Follow all symbolic links, unlike **−r**.

Other Options

−−line−buffered
Use line buffering on output. This can cause a performance penalty.

−−mmap
If possible, use the **mmap**(2) system call to read input, instead of the default **read**(2) system call. In some situations, **−−mmap** yields better performance. However, **−−mmap** can cause undefined behavior (including core dumps) if an input file shrinks while **grep** is operating, or if an I/O error occurs.

−U, **−−binary**
Treat the file(s) as binary. By default, under MS-DOS and MS-Windows, **grep** guesses the file type by looking at the contents of the first 32KB read from the file. If **grep** decides the file is a text file, it strips the CR characters from the original file contents (to make regular expressions with **^** and **$** work correctly). Specifying **−U** overrules this guesswork, causing all files to be read and passed to the matching mechanism verbatim; if the file is a text file with CR/LF pairs at the end of each line, this will cause some regular expressions to fail. This option has no effect on platforms other than MS-DOS and MS-Windows.

−z, **−−null−data**
Treat the input as a set of lines, each terminated by a zero byte (the ASCII **NUL** character) instead of a newline. Like the **-Z** or **−−null** option, this option can be used with commands like **sort -z** to process arbitrary file names.

REGULAR EXPRESSIONS

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions.

grep understands three different versions of regular expression syntax: “basic” (BRE), “extended” (ERE) and “perl” (PRCE). In GNU **grep**, there is no difference in available functionality between basic and extended syntaxes. In other implementations, basic regular expressions are less powerful. The following description applies to extended regular expressions; differences for basic regular expressions are summarized afterwards. Perl regular expressions give additional functionality, and are documented in pcresyntax(3) and pcrepattern(3), but only work if pcre is available in the system.

The fundamental building blocks are the regular expressions that match a single character. Most characters, including all letters and digits, are regular expressions that match themselves. Any meta-character with special meaning may be quoted by preceding it with a backslash.

The period **.** matches any single character.

Character Classes and Bracket Expressions

A *bracket expression* is a list of characters enclosed by **[** and **]**. It matches any single character in that list; if the first character of the list is the caret **^** then it matches any character *not* in the list. For example, the regular expression **[0123456789]** matches any single digit.

Within a bracket expression, a *range expression* consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters, inclusive, using the locale’s collating sequence and character set. For example, in the default C locale, **[a−d]** is equivalent to **[abcd]**. Many locales sort characters in dictionary order, and in these locales **[a−d]** is typically not equivalent to **[abcd]**; it might be equivalent to **[aBbCcDd]**, for example. To obtain the traditional interpretation of bracket expressions, you can use the C locale by setting the **LC_ALL** environment variable to the value **C**.

Finally, certain named classes of characters are predefined within bracket expressions, as follows. Their names are self explanatory, and they are **[:alnum:]**, **[:alpha:]**, **[:cntrl:]**, **[:digit:]**, **[:graph:]**, **[:lower:]**, **[:print:]**, **[:punct:]**, **[:space:]**, **[:upper:]**, and **[:xdigit:].** For example, **[[:alnum:]]** means the character class of numbers and letters in the current locale. In the C locale and ASCII character set encoding, this is the same as **[0−9A−Za−z]**. (Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket expression.) Most meta-characters lose their special meaning inside bracket expressions. To include a literal **]** place it first in the list. Similarly, to include a literal **^** place it anywhere but first. Finally, to include a literal **−** place it last.

Anchoring

The caret **^** and the dollar sign **$** are meta-characters that respectively match the empty string at the beginning and end of a line.

The Backslash Character and Special Expressions

The symbols \< and \> respectively match the empty string at the beginning and end of a word. The symbol **\b** matches the empty string at the edge of a word, and **\B** matches the empty string provided it’s *not* at the edge of a word. The symbol **\w** is a synonym for **[_[:alnum:]]** and **\W** is a synonym for **[^_[:alnum:]]**.

Repetition

A regular expression may be followed by one of several repetition operators:

? The preceding item is optional and matched at most once.
***** The preceding item will be matched zero or more times.
+ The preceding item will be matched one or more times.
{*n*} The preceding item is matched exactly *n* times.
{*n*,} The preceding item is matched *n* or more times.
{,*m*} The preceding item is matched at most *m* times. This is a GNU extension.
{*n*,*m*} The preceding item is matched at least *n* times, but not more than *m* times.

Concatenation

Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated expressions.

Alternation
Two regular expressions may be joined by the infix operator |; the resulting regular expression matches any string matching either alternate expression.

Precedence
Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A whole expression may be enclosed in parentheses to override these precedence rules and form a subexpression.

Back References and Subexpressions
The back-reference *n*, where *n* is a single digit, matches the substring previously matched by the *n*th parenthesized subexpression of the regular expression.

Basic vs Extended Regular Expressions
In basic regular expressions the meta-characters **?**, **+**, **{**, |, **(**, and **)** lose their special meaning; instead use the backslashed versions **\?**, **\+**, **\{**, \|, **\(**, and **\)**.

Traditional **egrep** did not support the **{** meta-character, and some **egrep** implementations support **\{** instead, so portable scripts should avoid **{** in **grep −E** patterns and should use **[{]** to match a literal **{**.

GNU **grep −E** attempts to support traditional usage by assuming that **{** is not special if it would be the start of an invalid interval specification. For example, the command **grep −E '{1'** searches for the two-character string **{1** instead of reporting a syntax error in the regular expression. POSIX allows this behavior as an extension, but portable scripts should avoid it.

ENVIRONMENT VARIABLES

The behavior of **grep** is affected by the following environment variables.

The locale for category **LC_***foo* is specified by examining the three environment variables **LC_ALL**, **LC_***foo*, **LANG**, in that order. The first of these variables that is set specifies the locale. For example, if **LC_ALL** is not set, but **LC_MESSAGES** is set to **pt_BR**, then the Brazilian Portuguese locale is used for the **LC_MESSAGES** category. The C locale is used if none of these environment variables are set, if the locale catalog is not installed, or if **grep** was not compiled with national language support (NLS).

GREP_OPTIONS
This variable specifies default options to be placed in front of any explicit options. For example, if **GREP_OPTIONS** is '**−−binary-files=without-match −−directories=skip**', **grep** behaves as if the two options **−−binary−files=without-match** and **−−directories=skip** had been specified before any explicit options. Option specifications are separated by whitespace. A backslash escapes the next character, so it can be used to specify an option containing whitespace or a backslash.

GREP_COLOR
This variable specifies the color used to highlight matched (non-empty) text. It is deprecated in favor of **GREP_COLORS**, but still supported. The **mt**, **ms**, and **mc** capabilities of **GREP_COLORS** have priority over it. It can only specify the color used to highlight the matching non-empty text in any matching line (a selected line when the **-v** command-line option is omitted, or a context line when **-v** is specified). The default is **01;31**, which means a bold red foreground text on the terminal's default background.

GREP_COLORS
Specifies the colors and other attributes used to highlight various parts of the output. Its value is a colon-separated list of capabilities that defaults to **ms=01;31:mc=01;31:sl=:cx=:fn=35:ln=32:bn=32:se=36** with the **rv** and **ne** boolean capabilities omitted (i.e., false). Supported capabilities are as follows.

sl= SGR substring for whole selected lines (i.e., matching lines when the **−v** command-line option is omitted, or non-matching lines when **−v** is specified). If however the boolean **rv** capability and the **−v** command-line option are both specified, it applies to context matching lines instead. The default is empty (i.e., the terminal's default color pair).

cx= SGR substring for whole context lines (i.e., non-matching lines when the **−v** command-line option is omitted, or matching lines when **−v** is specified). If however the boolean **rv**

capability and the **−v** command-line option are both specified, it applies to selected non-matching lines instead. The default is empty (i.e., the terminal's default color pair).

rv Boolean value that reverses (swaps) the meanings of the **sl=** and **cx=** capabilities when the **−v** command-line option is specified. The default is false (i.e., the capability is omitted).

mt=01;31
SGR substring for matching non-empty text in any matching line (i.e., a selected line when the **−v** command-line option is omitted, or a context line when **−v** is specified). Setting this is equivalent to setting both **ms=** and **mc=** at once to the same value. The default is a bold red text foreground over the current line background.

ms=01;31
SGR substring for matching non-empty text in a selected line. (This is only used when the **−v** command-line option is omitted.) The effect of the **sl=** (or **cx=** if **rv**) capability remains active when this kicks in. The default is a bold red text foreground over the current line background.

mc=01;31
SGR substring for matching non-empty text in a context line. (This is only used when the **−v** command-line option is specified.) The effect of the **cx=** (or **sl=** if **rv**) capability remains active when this kicks in. The default is a bold red text foreground over the current line background.

fn=35 SGR substring for file names prefixing any content line. The default is a magenta text foreground over the terminal's default background.

ln=32 SGR substring for line numbers prefixing any content line. The default is a green text foreground over the terminal's default background.

bn=32 SGR substring for byte offsets prefixing any content line. The default is a green text foreground over the terminal's default background.

se=36 SGR substring for separators that are inserted between selected line fields (**:**), between context line fields, (**−**), and between groups of adjacent lines when nonzero context is specified (**−−**). The default is a cyan text foreground over the terminal's default background.

ne Boolean value that prevents clearing to the end of line using Erase in Line (EL) to Right (**\33[K**) each time a colorized item ends. This is needed on terminals on which EL is not supported. It is otherwise useful on terminals for which the **back_color_erase** (**bce**) boolean terminfo capability does not apply, when the chosen highlight colors do not affect the background, or when EL is too slow or causes too much flicker. The default is false (i.e., the capability is omitted).

Note that boolean capabilities have no =... part. They are omitted (i.e., false) by default and become true when specified.

See the Select Graphic Rendition (SGR) section in the documentation of the text terminal that is used for permitted values and their meaning as character attributes. These substring values are integers in decimal representation and can be concatenated with semicolons. **grep** takes care of assembling the result into a complete SGR sequence (**\33[**...**m**). Common values to concatenate include **1** for bold, **4** for underline, **5** for blink, **7** for inverse, **39** for default foreground color, **30** to **37** for foreground colors, **90** to **97** for 16-color mode foreground colors, **38;5;0** to **38;5;255** for 88-color and 256-color modes foreground colors, **49** for default background color, **40** to **47** for background colors, **100** to **107** for 16-color mode background colors, and **48;5;0** to **48;5;255** for 88-color and 256-color modes background colors.

LC_ALL, **LC_COLLATE**, **LANG**
These variables specify the locale for the **LC_COLLATE** category, which determines the collating sequence used to interpret range expressions like **[a−z]**.

LC_ALL, **LC_CTYPE**, **LANG**
These variables specify the locale for the **LC_CTYPE** category, which determines the type of characters, e.g., which characters are whitespace.

LC_ALL, **LC_MESSAGES**, **LANG**
These variables specify the locale for the **LC_MESSAGES** category, which determines the language that **grep** uses for messages. The default C locale uses American English messages.

POSIXLY_CORRECT
If set, **grep** behaves as POSIX requires; otherwise, **grep** behaves more like other GNU programs. POSIX requires that options that follow file names must be treated as file names; by default, such options are permuted to the front of the operand list and are treated as options. Also, POSIX requires that unrecognized options be diagnosed as "illegal", but since they are not really against the law the default is to diagnose them as "invalid". **POSIXLY_CORRECT** also disables **_*N*_GNU_nonoption_argv_flags_**, described below.

_*N*_GNU_nonoption_argv_flags_
(Here *N* is **grep**'s numeric process ID.) If the *i*th character of this environment variable's value is **1**, do not consider the *i*th operand of **grep** to be an option, even if it appears to be one. A shell can put this variable in the environment for each command it runs, specifying which operands are the results of file name wildcard expansion and therefore should not be treated as options. This behavior is available only with the GNU C library, and only when **POSIXLY_CORRECT** is not set.

EXIT STATUS

The exit status is 0 if selected lines are found, and 1 if not found. If an error occurred the exit status is 2. (Note: POSIX error handling code should check for '2' or greater.)

COPYRIGHT

BUGS

Reporting Bugs

Email bug reports to <**bug−grep@gnu.org**>, a mailing list whose web page is <**http://lists.gnu.org/mailman/listinfo/bug−grep**>. **grep**'s Savannah bug tracker is located at <**http://savannah.gnu.org/bugs/?group=grep**>.

Known Bugs

Large repetition counts in the **{*n*,*m*}** construct may cause **grep** to use lots of memory. In addition, certain other obscure regular expressions require exponential time and space, and may cause **grep** to run out of memory.

Back-references are very slow, and may require exponential time.

SEE ALSO

Regular Manual Pages

awk(1), cmp(1), diff(1), find(1), gzip(1), perl(1), sed(1), sort(1), xargs(1), zgrep(1), mmap(2), read(2), pcre(3), pcresyntax(3), pcrepattern(3), terminfo(5), glob(7), regex(7).

POSIX Programmer's Manual Page

grep(1p).

TEXinfo Documentation

The full documentation for **grep** is maintained as a TEXinfo manual, which you can read at http://www.gnu.org/software/grep/manual/. If the **info** and **grep** programs are properly installed at your site, the command

info grep

should give you access to the complete manual.

NOTES

This man page is maintained only fitfully; the full documentation is often more up-to-date.

GNU's not Unix, but Unix is a beast; its plural form is Unixen.