



**NAME**

grep, egrep, fgrep – 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, two variant programs **egrep** and **fgrep** are available. **egrep** is the same as **grep -E**. **fgrep** is the same as **grep -F**. 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. 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 for tools like **zgrep**, e.g., **gzip -cd foo.gz | grep --label=foo something**

**-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 -IZ** 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**, which means that directories are read just as if they were ordinary files. If *ACTION* is **skip**, directories are silently skipped. If *ACTION* is **recurse**, **grep** reads all files under each directory, recursively; 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, -r, --recursive**  
Read all files under each directory, recursively; this is equivalent to the **-d recurse** option.

### 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 two different versions of regular expression syntax: “basic” and “extended.” In GNU **grep**, there is no difference in available functionality using either syntax. 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.

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 [**0-9A-Za-z**], except the latter form depends upon the C locale and the ASCII character encoding, whereas the former is independent of locale and character set. (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.
- {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.2 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.2 requires; otherwise, **grep** behaves more like other GNU programs. POSIX.2 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.2 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**

Normally, the exit status is 0 if selected lines are found and 1 otherwise. But the exit status is 2 if an error occurred, unless the **-q** or **--quiet** or **--silent** option is used and a selected line is found. Note, however, that POSIX only mandates, for programs such as **grep**, **cmp**, and **diff**, that the exit status in

case of error be greater than 1; it is therefore advisable, for the sake of portability, to use logic that tests for this general condition instead of strict equality with 2.

## COPYRIGHT

Copyright © 1998, 1999, 2000, 2002, 2005 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

## BUGS

### Reporting Bugs

Email bug reports to [<bug-grep@gnu.org>](mailto:bug-grep@gnu.org), a mailing list whose web page is [<http://lists.gnu.org/mailman/listinfo/bug-grep>](http://lists.gnu.org/mailman/listinfo/bug-grep). **grep**'s Savannah bug tracker is located at [<http://savannah.gnu.org/bugs/?group=grep>](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\(3\)](#), [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. 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

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

**PROLOG**

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

**NAME**

grep – search a file for a pattern

**SYNOPSIS**

```
grep [-E| -F][ -c| -l -q][ -insvx] -e pattern_list...
      [-f pattern_file]...[file...]
```

```
grep [-E| -F][ -c| -l -q][ -insvx][ -e pattern_list]...
      -f pattern_file...[file...]
```

```
grep [-E| -F][ -c| -l -q][ -insvx] pattern_list[file...]
```

**DESCRIPTION**

The *grep* utility shall search the input files, selecting lines matching one or more patterns; the types of patterns are controlled by the options specified. The patterns are specified by the **-e** option, **-f** option, or the *pattern\_list* operand. The *pattern\_list*'s value shall consist of one or more patterns separated by <newline>s; the *pattern\_file*'s contents shall consist of one or more patterns terminated by <newline>. By default, an input line shall be selected if any pattern, treated as an entire basic regular expression (BRE) as described in the Base Definitions volume of IEEE Std 1003.1-2001, Section 9.3, Basic Regular Expressions, matches any part of the line excluding the terminating <newline>; a null BRE shall match every line. By default, each selected input line shall be written to the standard output.

Regular expression matching shall be based on text lines. Since a <newline> separates or terminates patterns (see the **-e** and **-f** options below), regular expressions cannot contain a <newline>. Similarly, since patterns are matched against individual lines (excluding the terminating <newline>s) of the input, there is no way for a pattern to match a <newline> found in the input.

**OPTIONS**

The *grep* utility shall conform to the Base Definitions volume of IEEE Std 1003.1-2001, Section 12.2, Utility Syntax Guidelines.

The following options shall be supported:

- E** Match using extended regular expressions. Treat each pattern specified as an ERE, as described in the Base Definitions volume of IEEE Std 1003.1-2001, Section 9.4, Extended Regular Expressions. If any entire ERE pattern matches some part of an input line excluding the terminating <newline>, the line shall be matched. A null ERE shall match every line.
- F** Match using fixed strings. Treat each pattern specified as a string instead of a regular expression. If an input line contains any of the patterns as a contiguous sequence of bytes, the line shall be matched. A null string shall match every line.
- c** Write only a count of selected lines to standard output.
- e *pattern\_list***

Specify one or more patterns to be used during the search for input. The application shall ensure that patterns in *pattern\_list* are separated by a <newline>. A null pattern can be specified by two adjacent <newline>s in *pattern\_list*. Unless the **-E** or **-F** option is also specified, each pattern shall be treated as a BRE, as described in the Base Definitions volume of IEEE Std 1003.1-2001, Section 9.3, Basic Regular Expressions. Multiple **-e** and **-f** options shall be accepted by the *grep* utility. All of the specified patterns shall be used when matching lines, but the order of evaluation is unspecified.

**-f *pattern\_file***

Read one or more patterns from the file named by the pathname *pattern\_file*. Patterns in *pattern\_file* shall be terminated by a <newline>. A null pattern can be specified by an empty line in *pattern\_file*. Unless the **-E** or **-F** option is also specified, each pattern shall be treated as a

BRE, as described in the Base Definitions volume of IEEE Std 1003.1-2001, Section 9.3, Basic Regular Expressions.

- i** Perform pattern matching in searches without regard to case; see the Base Definitions volume of IEEE Std 1003.1-2001, Section 9.2, Regular Expression General Requirements.
- l** (The letter ell.) Write only the names of files containing selected lines to standard output. Pathnames shall be written once per file searched. If the standard input is searched, a pathname of "**standard input**" shall be written, in the POSIX locale. In other locales, "**standard input**" may be replaced by something more appropriate in those locales.
- n** Precede each output line by its relative line number in the file, each file starting at line 1. The line number counter shall be reset for each file processed.
- q** Quiet. Nothing shall be written to the standard output, regardless of matching lines. Exit with zero status if an input line is selected.
- s** Suppress the error messages ordinarily written for nonexistent or unreadable files. Other error messages shall not be suppressed.
- v** Select lines not matching any of the specified patterns. If the **-v** option is not specified, selected lines shall be those that match any of the specified patterns.
- x** Consider only input lines that use all characters in the line excluding the terminating <new-line> to match an entire fixed string or regular expression to be matching lines.

## OPERANDS

The following operands shall be supported:

*pattern\_list*

Specify one or more patterns to be used during the search for input. This operand shall be treated as if it were specified as **-e** *pattern\_list*.

*file*

A pathname of a file to be searched for the patterns. If no *file* operands are specified, the standard input shall be used.

## STDIN

The standard input shall be used only if no *file* operands are specified. See the INPUT FILES section.

## INPUT FILES

The input files shall be text files.

## ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of *grep*:

**LANG** Provide a default value for the internationalization variables that are unset or null. (See the Base Definitions volume of IEEE Std 1003.1-2001, Section 8.2, Internationalization Variables for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL**

If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_COLLATE**

Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements within regular expressions.

**LC\_CTYPE**

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files) and the behavior of character classes within regular expressions.

**LC\_MESSAGES**

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**NLSPATH**

Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

If the **-l** option is in effect, and the **-q** option is not, the following shall be written for each file containing at least one selected input line:

`"%s\n", <file>`

Otherwise, if more than one *file* argument appears, and **-q** is not specified, the *grep* utility shall prefix each output line by:

`"%s:", <file>`

The remainder of each output line shall depend on the other options specified:

- \* If the **-c** option is in effect, the remainder of each output line shall contain:

`"%d\n", <count>`

- \* Otherwise, if **-c** is not in effect and the **-n** option is in effect, the following shall be written to standard output:

`"%d:", <line number>`

- \* Finally, the following shall be written to standard output:

`"%s", <selected-line contents>`

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

- |    |                                  |
|----|----------------------------------|
| 0  | One or more lines were selected. |
| 1  | No lines were selected.          |
| >1 | An error occurred.               |

**CONSEQUENCES OF ERRORS**

If the **-q** option is specified, the exit status shall be zero if an input line is selected, even if an error was detected. Otherwise, default actions shall be performed.

*The following sections are informative.*

**APPLICATION USAGE**

Care should be taken when using characters in *pattern\_list* that may also be meaningful to the command interpreter. It is safest to enclose the entire *pattern\_list* argument in single quotes:

'...'

The **-e** *pattern\_list* option has the same effect as the *pattern\_list* operand, but is useful when *pattern\_list* begins with the hyphen delimiter. It is also useful when it is more convenient to provide multiple patterns as separate arguments.

Multiple **-e** and **-f** options are accepted and *grep* uses all of the patterns it is given while matching input text lines. (Note that the order of evaluation is not specified. If an implementation finds a null string as a pattern, it is allowed to use that pattern first, matching every line, and effectively ignore any other patterns.)

The **-q** option provides a means of easily determining whether or not a pattern (or string) exists in a group of files. When searching several files, it provides a performance improvement (because it can quit as soon as it finds the first match) and requires less care by the user in choosing the set of files to supply as arguments (because it exits zero if it finds a match even if *grep* detected an access or read error on earlier *file* operands).

## EXAMPLES

1. To find all uses of the word "Posix" (in any case) in file **text.mm** and write with line numbers:

```
grep -i -n posix text.mm
```

2. To find all empty lines in the standard input:

```
grep ^$
```

or:

```
grep -v .
```

3. Both of the following commands print all lines containing strings "abc" or "def" or both:

```
grep -E 'abc|def'
```

```
grep -F 'abc  
def'
```

4. Both of the following commands print all lines matching exactly "abc" or "def" :

```
grep -E '^abc$|^def$'
```

```
grep -F -x 'abc  
def'
```

## RATIONALE

This *grep* has been enhanced in an upwards-compatible way to provide the exact functionality of the historical *egrep* and *fgrep* commands as well. It was the clear intention of the standard developers to consolidate the three *greps* into a single command.

The old *egrep* and *fgrep* commands are likely to be supported for many years to come as implementation extensions, allowing historical applications to operate unmodified.

Historical implementations usually silently ignored all but one of multiply-specified **-e** and **-f** options, but were not consistent as to which specification was actually used.

The **-b** option was omitted from the OPTIONS section because block numbers are implementation-defined.

The System V restriction on using `-` to mean standard input was omitted.

A definition of action taken when given a null BRE or ERE is specified. This is an error condition in some historical implementations.

The `-I` option previously indicated that its use was undefined when no files were explicitly named. This behavior was historical and placed an unnecessary restriction on future implementations. It has been removed.

The historical BSD `grep -s` option practice is easily duplicated by redirecting standard output to `/dev/null`. The `-s` option required here is from System V.

The `-x` option, historically available only with `fgrep`, is available here for all of the non-obsolete versions.

## **FUTURE DIRECTIONS**

None.

## **SEE ALSO**

*sed*

## **COPYRIGHT**

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at <http://www.open-group.org/unix/online.html> .