GREP(1)   GREP(1)

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 FILEs (or standard input if no files are
       named, or the file name - is given) 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.

OPTIONS

       -A NUM, --after-context=NUM
          Print NUM lines of trailing context after matching lines.
          Places a line containing -- between contiguous groups of
          matches.

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

       -B NUM, --before-context=NUM
          Print NUM lines of leading context before matching lines.
          Places a line containing -- between contiguous groups of
          matches.

       -C NUM, --context=NUM
          Print NUM lines of output context. Places a line containing --

       between contiguous groups of matches.

       -b, --byte-offset
          Print the byte offset within the input file before each line of
          output.

       --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 mes-
          sage 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 termi-
          nal driver interprets some of it as commands.

       --colour[=WHEN], --color[=WHEN]
          Surround the matching string with the marker find in GREP_COLOR
          environment variable. WHEN may be 'never', 'always', or 'auto'

       -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.

       -D ACTION, --devices=ACTION
          If an input file is a device, FIFO or socket, use ACTION to pro-
          cess 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, direc-
          tories are silently skipped. If ACTION is recurse, grep reads
          all files under each directory, recursively; this is equivalent
          to the -r option.

-E, --extended-regexp
Interpret PATTERN as an extended regular expression (see below).

-e PATTERN, --regexp=PATTERN
Use PATTERN as the pattern; useful to protect patterns beginning
with -.

-F, --fixed-strings
Interpret PATTERN as a list of fixed strings, separated by new-
lines, any of which is to be matched.

-P, --perl-regexp
Interpret PATTERN as a Perl regular expression.

-f FILE, --file=FILE
Obtain patterns from FILE, one per line. The empty file con-
tains zero patterns, and therefore matches nothing.

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

-H, --with-filename
Print the filename for each match.

-h, --no-filename
Suppress the prefixing of filenames on output when multiple
files are searched.

--help Output a brief help message.

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

-i, --ignore-case
Ignore case distinctions in both the PATTERN and the input
files.

-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.

-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.

--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.

-n, --line-number
Prefix each line of output with the line number within its input
file.

-o, --only-matching
Show only the part of a matching line that matches PATTERN.

--label=LABEL
Displays input actually coming from standard input as input com-
ing from file LABEL. This is especially useful for tools like
zgrep, e.g. gzip -cd foo.gz |grep -H --label=foo something

--line-buffered
Use line buffering, it can be a performance penality.

-q, --quiet, --silent
Quiet; do not write anything to standard output. Exit immedi-
ately with zero status if any match is found, even if an error

was detected. Also see the -s or --no-messages option.

-R, -r, --recursive
    Read all files under each directory, recursively; this is equiv-
    alent to the -d recurse option.

--include=PATTERN
    Recurse in directories only searching file matching PATTERN.

--exclude=PATTERN
    Recurse in directories skip file matching PATTERN.

-s, --no-messages
    Suppress error messages about nonexistent or unreadable files.
    Portability note: unlike GNU grep, traditional grep did not con-
    form to POSIX.2, because traditional grep lacked a -q option and
    its -s option behaved like GNU grep's -q option. Shell scripts
    intended to be portable to traditional grep should avoid both -q
    and -s and should redirect output to /dev/null instead.

-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-Win-
    dows.

-u, --unix-byte-offsets
    Report Unix-style byte offsets.   This switch causes grep to
    report byte offsets as if the file were 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.

-V, --version

    Print the version number of grep to standard error. This ver-
    sion number should be included in all bug reports (see below).

-v, --invert-match
    Invert the sense of matching, to select non-matching lines.

-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 con-
    stituent 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.

-y   Obsolete synonym for -i.

-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.

REGULAR EXPRESSIONS

    A regular expression is a pattern that describes a set of strings.
    Regular expressions are constructed analogously to arithmetic expres-
    sions, 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 avail-
    able 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 metacharacter with
special meaning may be quoted by preceding it with a backslash.

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 charac-
ters 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 dictio-
nary 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 list.)
Most metacharacters lose their special meaning inside lists.   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.

The period . matches any single character. The symbol \w is a synonym
for [[:alnum:]] and \W is a synonym for [^[:alnum]].

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

A regular expression may be followed by one of several repetition oper-
ators:
?   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.
{n,m}   The preceding item is matched at least n times, but not more
   than m times.

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

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

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

The backreference \n, where n is a single digit, matches the substring
previously matched by the nth parenthesized subexpression of the regu-
lar expression.

In basic regular expressions the metacharacters ?, +, {, |, (, and)
lose their special meaning; instead use the backslashed versions \?,
\+, \{, \|, \(, and \).

Traditional egrep did not support the { metacharacter, and some egrep
implementations support \{ instead, so portable scripts should avoid {
in egrep patterns and should use [{] to match a literal {.

GNU egrep attempts to support traditional usage by assuming that { is
not special if it would be the start of an invalid interval specifica-
tion.   For example, the shell command egrep '{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

Grep's behavior is affected by the following environment variables.

A locale 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 Brazilian Portuguese is
used for the LC_MESSAGES locale. The C locale is used if none of these
environment variables are set, or 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 --direc-
      tories=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
      Specifies the marker for highlighting.

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

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

   LC_ALL, LC_MESSAGES, LANG
      These variables specify the LC_MESSAGES locale, 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_nonop-
      tion_argv_flags_, described below.

   _N_GNU_nonoption_argv_flags_
      (Here N is grep's numeric process ID.) If the ith character of
      this environment variable's value is 1, do not consider the ith
      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 com-
      mand 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.

DIAGNOSTICS

Normally, 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.

BUGS

Email bug reports to bug-grep@gnu.org.

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.

Backreferences are very slow, and may require exponential time.