NAME

pcre2grep - a grep with Perl-compatible regular expressions.

SYNOPSIS

pcre2grep [options] [long options] [pattern] [path1 path2 ...]

DESCRIPTION

pcre2grep searches files for character patterns, in the same way as other grep commands do, but it uses the PCRE2 regular expression library to support patterns that are compatible with the regular expressions of Perl 5. See **pcre2syntax**(3) for a quick-reference summary of pattern syntax, or **pcre2pattern**(3) for a full description of the syntax and semantics of the regular expressions that PCRE2 supports.

Patterns, whether supplied on the command line or in a separate file, are given without delimiters. For example:

```
pcre2grep Thursday /etc/motd
```

If you attempt to use delimiters (for example, by surrounding a pattern with slashes, as is common in Perl scripts), they are interpreted as part of the pattern. Quotes can of course be used to delimit patterns on the command line because they are interpreted by the shell, and indeed quotes are required if a pattern contains white space or shell metacharacters.

The first argument that follows any option settings is treated as the single pattern to be matched when neither **-e** nor **-f** is present. Conversely, when one or both of these options are used to specify patterns, all arguments are treated as path names. At least one of **-e**, **-f**, or an argument pattern must be provided.

If no files are specified, **pcre2grep** reads the standard input. The standard input can also be referenced by a name consisting of a single hyphen. For example:

```
pcre2grep some-pattern file1 - file3
```

Input files are searched line by line. By default, each line that matches a pattern is copied to the standard output, and if there is more than one file, the file name is output at the start of each line, followed by a colon. However, there are options that can change how **pcre2grep** behaves. In particular, the **-M** option makes it possible to search for strings that span line boundaries. What defines a line boundary is controlled by the **-N** (**--newline**) option.

The amount of memory used for buffering files that are being scanned is controlled by parameters that can be set by the **--buffer-size** and **--max-buffer-size** options. The first of these sets the size of buffer that is obtained at the start of processing. If an input file contains very long lines, a larger buffer may be needed; this is handled by automatically extending the buffer, up to the limit specified by **--max-buffer-size**. The default values for these parameters can be set when **pcre2grep** is built; if nothing is specified, the defaults are set to 20KiB and 1MiB respectively. An error occurs if a line is too long and the buffer can no longer be expanded.

The block of memory that is actually used is three times the "buffer size", to allow for buffering "before" and "after" lines. If the buffer size is too small, fewer than requested "before" and "after" lines may be output.

Patterns can be no longer than 8KiB or BUFSIZ bytes, whichever is the greater. BUFSIZ is defined in **<stdio.h>**. When there is more than one pattern (specified by the use of **-e** and/or **-f**), each pattern is applied to each line in the order in which they are defined, except that all the **-e** patterns are tried before the **-f** patterns.

By default, as soon as one pattern matches a line, no further patterns are considered. However, if **--colour** (or **--color**) is used to colour the matching substrings, or if **--only-matching**, **--file-offsets**, or **--line-offsets** is used to output only the part of the line that matched (either shown literally, or as an offset), scanning resumes immediately following the match, so that further matches on the same line can be found. If there are multiple patterns, they are all tried on the remainder of the line, but patterns that follow the one that

matched are not tried on the earlier part of the line.

This behaviour means that the order in which multiple patterns are specified can affect the output when one of the above options is used. This is no longer the same behaviour as GNU grep, which now manages to display earlier matches for later patterns (as long as there is no overlap).

Patterns that can match an empty string are accepted, but empty string matches are never recognized. An example is the pattern "(super)?(man)?", in which all components are optional. This pattern finds all occurrences of both "super" and "man"; the output differs from matching with "super|man" when only the matching substrings are being shown.

If the **LC_ALL** or **LC_CTYPE** environment variable is set, **pcre2grep** uses the value to set a locale when calling the PCRE2 library. The **--locale** option can be used to override this.

SUPPORT FOR COMPRESSED FILES

It is possible to compile **pcre2grep** so that it uses **libz** or **libbz2** to read compressed files whose names end in **.gz** or **.bz2**, respectively. You can find out whether your **pcre2grep** binary has support for one or both of these file types by running it with the **--help** option. If the appropriate support is not present, all files are treated as plain text. The standard input is always so treated. When input is from a compressed .gz or .bz2 file, the **--line-buffered** option is ignored.

BINARY FILES

By default, a file that contains a binary zero byte within the first 1024 bytes is identified as a binary file, and is processed specially. (GNU grep identifies binary files in this manner.) However, if the newline type is specified as "nul", that is, the line terminator is a binary zero, the test for a binary file is not applied. See the **--binary-files** option for a means of changing the way binary files are handled.

BINARY ZEROS IN PATTERNS

Patterns passed from the command line are strings that are terminated by a binary zero, so cannot contain internal zeros. However, patterns that are read from a file via the **-f** option may contain binary zeros.

OPTIONS

The order in which some of the options appear can affect the output. For example, both the **-H** and **-l** options affect the printing of file names. Whichever comes later in the command line will be the one that takes effect. Similarly, except where noted below, if an option is given twice, the later setting is used. Numerical values for options may be followed by K or M, to signify multiplication by 1024 or 1024*1024 respectively.

-- This terminates the list of options. It is useful if the next item on the command line starts with a hyphen but is not an option. This allows for the processing of patterns and file names that start with hyphens.

-A *number*, **--after-context=***number*
Output up to *number* lines of context after each matching line. Fewer lines are output if the next match or the end of the file is reached, or if the processing buffer size has been set too small. If file names and/or line numbers are being output, a hyphen separator is used instead of a colon for the context lines. A line containing "--" is output between each group of lines, unless they are in fact contiguous in the input file. The value of *number* is expected to be relatively small. When **-c** is used, **-A** is ignored.

-a, **--text** Treat binary files as text. This is equivalent to **--binary-files=***text*.

-B *number*, **--before-context=***number*
Output up to *number* lines of context before each matching line. Fewer lines are output if the previous match or the start of the file is within *number* lines, or if the processing buffer size has been set too small. If file names and/or line numbers are being output, a hyphen separator is used instead of a colon for the context lines. A line containing "--" is output between each

group of lines, unless they are in fact contiguous in the input file. The value of *number* is expected to be relatively small. When **-c** is used, **-B** is ignored.

--binary-files=*word*

Specify how binary files are to be processed. If the word is "binary" (the default), pattern matching is performed on binary files, but the only output is "Binary file <name> matches" when a match succeeds. If the word is "text", which is equivalent to the **-a** or **--text** option, binary files are processed in the same way as any other file. In this case, when a match succeeds, the output may be binary garbage, which can have nasty effects if sent to a terminal. If the word is "without-match", which is equivalent to the **-I** option, binary files are not processed at all; they are assumed not to be of interest and are skipped without causing any output or affecting the return code.

--buffer-size=*number*

Set the parameter that controls how much memory is obtained at the start of processing for buffering files that are being scanned. See also **--max-buffer-size** below.

-C *number*, **--context=***number*

Output *number* lines of context both before and after each matching line. This is equivalent to setting both **-A** and **-B** to the same value.

-c, **--count** Do not output lines from the files that are being scanned; instead output the number of lines that would have been shown, either because they matched, or, if **-v** is set, because they failed to match. By default, this count is exactly the same as the number of lines that would have been output, but if the **-M** (multiline) option is used (without **-v**), there may be more suppressed lines than the count (that is, the number of matches).

If no lines are selected, the number zero is output. If several files are are being scanned, a count is output for each of them and the **-t** option can be used to cause a total to be output at the end. However, if the **--files-with-matches** option is also used, only those files whose counts are greater than zero are listed. When **-c** is used, the **-A**, **-B**, and **-C** options are ignored.

--colour, **--color**

If this option is given without any data, it is equivalent to "--colour=auto". If data is required, it must be given in the same shell item, separated by an equals sign.

--colour=*value*, **--color=***value*

This option specifies under what circumstances the parts of a line that matched a pattern should be coloured in the output. By default, the output is not coloured. The value (which is optional, see above) may be "never", "always", or "auto". In the latter case, colouring happens only if the standard output is connected to a terminal. More resources are used when colouring is enabled, because **pcre2grep** has to search for all possible matches in a line, not just one, in order to colour them all.

The colour that is used can be specified by setting one of the environment variables PCRE2GREP_COLOUR, PCRE2GREP_COLOR, PCREGREP_COLOUR, or PCREGREP_COLOR, which are checked in that order. If none of these are set, **pcre2grep** looks for GREP_COLORS or GREP_COLOR (in that order). The value of the variable should be a string of two numbers, separated by a semicolon, except in the case of GREP_COLORS, which must start with "ms=" or "mt=" followed by two semicolon-separated colours, terminated by the end of the string or by a colon. If GREP_COLORS does not start with "ms=" or "mt=" it is ignored, and GREP_COLOR is checked.

If the string obtained from one of the above variables contains any characters other than semicolon or digits, the setting is ignored and the default colour is used. The string is copied directly into the control string for setting colour on a terminal, so it is your responsibility to ensure that the values make sense. If no relevant environment variable is set, the default is "1;31", which gives red.

-D *action*, **--devices=***action*

If an input path is not a regular file or a directory, "action" specifies how it is to be processed. Valid values are "read" (the default) or "skip" (silently skip the path).

-d *action*, **--directories=***action*

If an input path is a directory, "action" specifies how it is to be processed. Valid values are "read" (the default in non-Windows environments, for compatibility with GNU grep), "recurse" (equivalent to the **-r** option), or "skip" (silently skip the path, the default in Windows environments). In the "read" case, directories are read as if they were ordinary files. In some operating systems the effect of reading a directory like this is an immediate end-of-file; in others it may provoke an error.

--depth-limit=*number*

See **--match-limit** below.

-e *pattern*, **--regex=***pattern*, **--regexp=***pattern*

Specify a pattern to be matched. This option can be used multiple times in order to specify several patterns. It can also be used as a way of specifying a single pattern that starts with a hyphen. When **-e** is used, no argument pattern is taken from the command line; all arguments are treated as file names. There is no limit to the number of patterns. They are applied to each line in the order in which they are defined until one matches.

If **-f** is used with **-e**, the command line patterns are matched first, followed by the patterns from the file(s), independent of the order in which these options are specified. Note that multiple use of **-e** is not the same as a single pattern with alternatives. For example, X|Y finds the first character in a line that is X or Y, whereas if the two patterns are given separately, with X first, **pcre2grep** finds X if it is present, even if it follows Y in the line. It finds Y only if there is no X in the line. This matters only if you are using **-o** or **--colo(u)r** to show the part(s) of the line that matched.

--exclude=*pattern*

Files (but not directories) whose names match the pattern are skipped without being processed. This applies to all files, whether listed on the command line, obtained from **--file-list**, or by scanning a directory. The pattern is a PCRE2 regular expression, and is matched against the final component of the file name, not the entire path. The **-F**, **-w**, and **-x** options do not apply to this pattern. The option may be given any number of times in order to specify multiple patterns. If a file name matches both an **--include** and an **--exclude** pattern, it is excluded. There is no short form for this option.

--exclude-from=*filename*

Treat each non-empty line of the file as the data for an **--exclude** option. What constitutes a newline when reading the file is the operating system's default. The **--newline** option has no effect on this option. This option may be given more than once in order to specify a number of files to read.

--exclude-dir=*pattern*

Directories whose names match the pattern are skipped without being processed, whatever the setting of the **--recursive** option. This applies to all directories, whether listed on the command line, obtained from **--file-list**, or by scanning a parent directory. The pattern is a PCRE2 regular expression, and is matched against the final component of the directory name, not the entire path. The **-F**, **-w**, and **-x** options do not apply to this pattern. The option may be given any number of times in order to specify more than one pattern. If a directory matches both **--include-dir** and **--exclude-dir**, it is excluded. There is no short form for this option.

-F, **--fixed-strings**

Interpret each data-matching pattern as a list of fixed strings, separated by newlines, instead of as a regular expression. What constitutes a newline for this purpose is controlled by the **--newline** option. The **-w** (match as a word) and **-x** (match whole line) options can be used with **-F**. They apply to each of the fixed strings. A line is selected if any of the fixed strings are found in

it (subject to **-w** or **-x**, if present). This option applies only to the patterns that are matched against the contents of files; it does not apply to patterns specified by any of the **--include** or **--exclude** options.

-f *filename*, **--file=***filename*
Read patterns from the file, one per line, and match them against each line of input. As is the case with patterns on the command line, no delimiters should be used. What constitutes a newline when reading the file is the operating system's default interpretation of \n. The **--newline** option has no effect on this option. Trailing white space is removed from each line, and blank lines are ignored. An empty file contains no patterns and therefore matches nothing. Patterns read from a file in this way may contain binary zeros, which are treated as ordinary data characters. See also the comments about multiple patterns versus a single pattern with alternatives in the description of **-e** above.

If this option is given more than once, all the specified files are read. A data line is output if any of the patterns match it. A file name can be given as "-" to refer to the standard input. When **-f** is used, patterns specified on the command line using **-e** may also be present; they are tested before the file's patterns. However, no other pattern is taken from the command line; all arguments are treated as the names of paths to be searched.

--file-list=*filename*
Read a list of files and/or directories that are to be scanned from the given file, one per line. What constitutes a newline when reading the file is the operating system's default. Trailing white space is removed from each line, and blank lines are ignored. These paths are processed before any that are listed on the command line. The file name can be given as "-" to refer to the standard input. If **--file** and **--file-list** are both specified as "-", patterns are read first. This is useful only when the standard input is a terminal, from which further lines (the list of files) can be read after an end-of-file indication. If this option is given more than once, all the specified files are read.

--file-offsets
Instead of showing lines or parts of lines that match, show each match as an offset from the start of the file and a length, separated by a comma. In this mode, no context is shown. That is, the **-A**, **-B**, and **-C** options are ignored. If there is more than one match in a line, each of them is shown separately. This option is mutually exclusive with **--output**, **--line-offsets**, and **--only-matching**.

-H, **--with-filename**
Force the inclusion of the file name at the start of output lines when searching a single file. By default, the file name is not shown in this case. For matching lines, the file name is followed by a colon; for context lines, a hyphen separator is used. If a line number is also being output, it follows the file name. When the **-M** option causes a pattern to match more than one line, only the first is preceded by the file name. This option overrides any previous **-h**, **-l**, or **-L** options.

-h, **--no-filename**
Suppress the output file names when searching multiple files. By default, file names are shown when multiple files are searched. For matching lines, the file name is followed by a colon; for context lines, a hyphen separator is used. If a line number is also being output, it follows the file name. This option overrides any previous **-H**, **-L**, or **-l** options.

--heap-limit=*number*
See **--match-limit** below.

--help Output a help message, giving brief details of the command options and file type support, and then exit. Anything else on the command line is ignored.

-I Ignore binary files. This is equivalent to **--binary-files=***without-match*.

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

Ignore upper/lower case distinctions during comparisons.

--include=*pattern*

If any **--include** patterns are specified, the only files that are processed are those that match one of the patterns (and do not match an **--exclude** pattern). This option does not affect directories, but it applies to all files, whether listed on the command line, obtained from **--file-list**, or by scanning a directory. The pattern is a PCRE2 regular expression, and is matched against the final component of the file name, not the entire path. The **-F**, **-w**, and **-x** options do not apply to this pattern. The option may be given any number of times. If a file name matches both an **--include** and an **--exclude** pattern, it is excluded. There is no short form for this option.

--include-from=*filename*

Treat each non-empty line of the file as the data for an **--include** option. What constitutes a newline for this purpose is the operating system's default. The **--newline** option has no effect on this option. This option may be given any number of times; all the files are read.

--include-dir=*pattern*

If any **--include-dir** patterns are specified, the only directories that are processed are those that match one of the patterns (and do not match an **--exclude-dir** pattern). This applies to all directories, whether listed on the command line, obtained from **--file-list**, or by scanning a parent directory. The pattern is a PCRE2 regular expression, and is matched against the final component of the directory name, not the entire path. The **-F**, **-w**, and **-x** options do not apply to this pattern. The option may be given any number of times. If a directory matches both **--include-dir** and **--exclude-dir**, it is excluded. There is no short form for this option.

-L, **--files-without-match**

Instead of outputting lines from the files, just output the names of the files that do not contain any lines that would have been output. Each file name is output once, on a separate line. This option overrides any previous **-H**, **-h**, or **-l** options.

-l, **--files-with-matches**

Instead of outputting lines from the files, just output the names of the files containing lines that would have been output. Each file name is output once, on a separate line. Searching normally stops as soon as a matching line is found in a file. However, if the **-c** (count) option is also used, matching continues in order to obtain the correct count, and those files that have at least one match are listed along with their counts. Using this option with **-c** is a way of suppressing the listing of files with no matches. This opeion overrides any previous **-H**, **-h**, or **-L** options.

--label=*name*

This option supplies a name to be used for the standard input when file names are being output. If not supplied, "(standard input)" is used. There is no short form for this option.

--line-buffered

When this option is given, non-compressed input is read and processed line by line, and the output is flushed after each write. By default, input is read in large chunks, unless **pcre2grep** can determine that it is reading from a terminal (which is currently possible only in Unix-like environments or Windows). Output to terminal is normally automatically flushed by the operating system. This option can be useful when the input or output is attached to a pipe and you do not want **pcre2grep** to buffer up large amounts of data. However, its use will affect performance, and the **-M** (multiline) option ceases to work. When input is from a compressed .gz or .bz2 file, **--line-buffered** is ignored.

--line-offsets

Instead of showing lines or parts of lines that match, show each match as a line number, the offset from the start of the line, and a length. The line number is terminated by a colon (as usual; see the **-n** option), and the offset and length are separated by a comma. In this mode, no context is shown. That is, the **-A**, **-B**, and **-C** options are ignored. If there is more than one match in a line, each of them is shown separately. This option is mutually exclusive with

--output, **--file-offsets**, and **--only-matching**.

--locale=*locale-name*

This option specifies a locale to be used for pattern matching. It overrides the value in the **LC_ALL** or **LC_CTYPE** environment variables. If no locale is specified, the PCRE2 library's default (usually the "C" locale) is used. There is no short form for this option.

--match-limit=*number*

Processing some regular expression patterns may take a very long time to search for all possible matching strings. Others may require a very large amount of memory. There are three options that set resource limits for matching.

The **--match-limit** option provides a means of limiting computing resource usage when processing patterns that are not going to match, but which have a very large number of possibilities in their search trees. The classic example is a pattern that uses nested unlimited repeats. Internally, PCRE2 has a counter that is incremented each time around its main processing loop. If the value set by **--match-limit** is reached, an error occurs.

The **--heap-limit** option specifies, as a number of kibibytes (units of 1024 bytes), the amount of heap memory that may be used for matching. Heap memory is needed only if matching the pattern requires a significant number of nested backtracking points to be remembered. This parameter can be set to zero to forbid the use of heap memory altogether.

The **--depth-limit** option limits the depth of nested backtracking points, which indirectly limits the amount of memory that is used. The amount of memory needed for each backtracking point depends on the number of capturing parentheses in the pattern, so the amount of memory that is used before this limit acts varies from pattern to pattern. This limit is of use only if it is set smaller than **--match-limit**.

There are no short forms for these options. The default limits can be set when the PCRE2 library is compiled; if they are not specified, the defaults are very large and so effectively unlimited.

--max-buffer-size=*number*

This limits the expansion of the processing buffer, whose initial size can be set by **--buffer-size**. The maximum buffer size is silently forced to be no smaller than the starting buffer size.

-M, **--multiline**

Allow patterns to match more than one line. When this option is set, the PCRE2 library is called in "multiline" mode. This allows a matched string to extend past the end of a line and continue on one or more subsequent lines. Patterns used with **-M** may usefully contain literal newline characters and internal occurrences of ^ and $ characters. The output for a successful match may consist of more than one line. The first line is the line in which the match started, and the last line is the line in which the match ended. If the matched string ends with a newline sequence, the output ends at the end of that line. If **-v** is set, none of the lines in a multi-line match are output. Once a match has been handled, scanning restarts at the beginning of the line after the one in which the match ended.

The newline sequence that separates multiple lines must be matched as part of the pattern. For example, to find the phrase "regular expression" in a file where "regular" might be at the end of a line and "expression" at the start of the next line, you could use this command:

```
pcre2grep -M 'regular\s+expression' <file>
```

The \s escape sequence matches any white space character, including newlines, and is followed by + so as to match trailing white space on the first line as well as possibly handling a two-character newline sequence.

There is a limit to the number of lines that can be matched, imposed by the way that **pcre2grep** buffers the input file as it scans it. With a sufficiently large processing buffer, this should not be a problem, but the **-M** option does not work when input is read line by line (see **--line-buffered**.)

-N *newline-type*, **--newline**=*newline-type*

The PCRE2 library supports five different conventions for indicating the ends of lines. They are the single-character sequences CR (carriage return) and LF (linefeed), the two-character sequence CRLF, an "anycrlf" convention, which recognizes any of the preceding three types, and an "any" convention, in which any Unicode line ending sequence is assumed to end a line. The Unicode sequences are the three just mentioned, plus VT (vertical tab, U+000B), FF (form feed, U+000C), NEL (next line, U+0085), LS (line separator, U+2028), and PS (paragraph separator, U+2029).

When the PCRE2 library is built, a default line-ending sequence is specified. This is normally the standard sequence for the operating system. Unless otherwise specified by this option, **pcre2grep** uses the library's default. The possible values for this option are CR, LF, CRLF, ANYCRLF, or ANY. This makes it possible to use **pcre2grep** to scan files that have come from other environments without having to modify their line endings. If the data that is being scanned does not agree with the convention set by this option, **pcre2grep** may behave in strange ways. Note that this option does not apply to files specified by the **-f**, **--exclude-from**, or **--include-from** options, which are expected to use the operating system's standard newline sequence.

-n, **--line-number**

Precede each output line by its line number in the file, followed by a colon for matching lines or a hyphen for context lines. If the file name is also being output, it precedes the line number. When the **-M** option causes a pattern to match more than one line, only the first is preceded by its line number. This option is forced if **--line-offsets** is used.

--no-jit If the PCRE2 library is built with support for just-in-time compiling (which speeds up matching), **pcre2grep** automatically makes use of this, unless it was explicitly disabled at build time. This option can be used to disable the use of JIT at run time. It is provided for testing and working round problems. It should never be needed in normal use.

-O *text*, **--output**=*text*

When there is a match, instead of outputting the whole line that matched, output just the given text. This option is mutually exclusive with **--only-matching**, **--file-offsets**, and **--line-offsets**. Escape sequences starting with a dollar character may be used to insert the contents of the matched part of the line and/or captured substrings into the text.

$<digits> or ${<digits>} is replaced by the captured substring of the given decimal number; zero substitutes the whole match. If the number is greater than the number of capturing substrings, or if the capture is unset, the replacement is empty.

$a is replaced by bell; $b by backspace; $e by escape; $f by form feed; $n by newline; $r by carriage return; $t by tab; $v by vertical tab.

$o<digits> is replaced by the character represented by the given octal number; up to three digits are processed.

$x<digits> is replaced by the character represented by the given hexadecimal number; up to two digits are processed.

Any other character is substituted by itself. In particular, $$ is replaced by a single dollar.

-o, **--only-matching**
Show only the part of the line that matched a pattern instead of the whole line. In this mode, no context is shown. That is, the **-A**, **-B**, and **-C** options are ignored. If there is more than one match in a line, each of them is shown separately, on a separate line of output. If **-o** is combined with **-v** (invert the sense of the match to find non-matching lines), no output is generated, but the return code is set appropriately. If the matched portion of the line is empty, nothing is output unless the file name or line number are being printed, in which case they are shown on an otherwise empty line. This option is mutually exclusive with **--output**, **--file-offsets** and **--line-offsets**.

-o*number*, **--only-matching**=*number*
Show only the part of the line that matched the capturing parentheses of the given number. Up to 50 capturing parentheses are supported by default. This limit can be changed via the **--om-capture** option. A pattern may contain any number of capturing parentheses, but only those whose number is within the limit can be accessed by **-o**. An error occurs if the number specified by **-o** is greater than the limit.

-o0 is the same as **-o** without a number. Because these options can be given without an argument (see above), if an argument is present, it must be given in the same shell item, for example, -o3 or --only-matching=2. The comments given for the non-argument case above also apply to this option. If the specified capturing parentheses do not exist in the pattern, or were not set in the match, nothing is output unless the file name or line number are being output.

If this option is given multiple times, multiple substrings are output for each match, in the order the options are given, and all on one line. For example, -o3 -o1 -o3 causes the substrings matched by capturing parentheses 3 and 1 and then 3 again to be output. By default, there is no separator (but see the next but one option).

--om-capture=*number*
Set the number of capturing parentheses that can be accessed by **-o**. The default is 50.

--om-separator=*text*
Specify a separating string for multiple occurrences of **-o**. The default is an empty string. Separating strings are never coloured.

-q, **--quiet** Work quietly, that is, display nothing except error messages. The exit status indicates whether or not any matches were found.

-r, **--recursive**
If any given path is a directory, recursively scan the files it contains, taking note of any **--include** and **--exclude** settings. By default, a directory is read as a normal file; in some operating systems this gives an immediate end-of-file. This option is a shorthand for setting the **-d** option to "recurse".

--recursion-limit=*number*
See **--match-limit** above.

-s, **--no-messages**
Suppress error messages about non-existent or unreadable files. Such files are quietly skipped. However, the return code is still 2, even if matches were found in other files.

-t, **--total-count**
This option is useful when scanning more than one file. If used on its own, **-t** suppresses all output except for a grand total number of matching lines (or non-matching lines if **-v** is used) in all the files. If **-t** is used with **-c**, a grand total is output except when the previous output is just one line. In other words, it is not output when just one file's count is listed. If file names are being output, the grand total is preceded by "TOTAL:". Otherwise, it appears as just another number. The **-t** option is ignored when used with **-L** (list files without matches), because the grand total would always be zero.

-u, **--utf** Operate in UTF-8 mode. This option is available only if PCRE2 has been compiled with UTF-8 support. All patterns (including those for any **--exclude** and **--include** options) and all subject lines that are scanned must be valid strings of UTF-8 characters.

-U, **--utf-allow-invalid**
As **--utf**, but in addition subject lines may contain invalid UTF-8 code unit sequences. These can never form part of any pattern match. This facility allows valid UTF-8 strings to be sought in executable or other binary files. For more details about matching in non-valid UTF-8 strings, see the **pcre2unicode**(3) documentation.

-V, **--version**
Write the version numbers of **pcre2grep** and the PCRE2 library to the standard output and then exit. Anything else on the command line is ignored.

-v, **--invert-match**
Invert the sense of the match, so that lines which do *not* match any of the patterns are the ones that are found.

-w, **--word-regex**, **--word-regexp**
Force the patterns only to match "words". That is, there must be a word boundary at the start and end of each matched string. This is equivalent to having "\b(?:" at the start of each pattern, and ")\b" at the end. This option applies only to the patterns that are matched against the contents of files; it does not apply to patterns specified by any of the **--include** or **--exclude** options.

-x, **--line-regex**, **--line-regexp**
Force the patterns to start matching only at the beginnings of lines, and in addition, require them to match entire lines. In multiline mode the match may be more than one line. This is equivalent to having "^(?:" at the start of each pattern and ")$" at the end. This option applies only to the patterns that are matched against the contents of files; it does not apply to patterns specified by any of the **--include** or **--exclude** options.

ENVIRONMENT VARIABLES

The environment variables **LC_ALL** and **LC_CTYPE** are examined, in that order, for a locale. The first one that is set is used. This can be overridden by the **--locale** option. If no locale is set, the PCRE2 library's default (usually the "C" locale) is used.

NEWLINES

The **-N** (**--newline**) option allows **pcre2grep** to scan files with different newline conventions from the default. Any parts of the input files that are written to the standard output are copied identically, with whatever newline sequences they have in the input. However, the setting of this option affects only the way scanned files are processed. It does not affect the interpretation of files specified by the **-f**, **--file-list**, **--exclude-from**, or **--include-from** options, nor does it affect the way in which **pcre2grep** writes informational messages to the standard error and output streams. For these it uses the string "\n" to indicate newlines, relying on the C I/O library to convert this to an appropriate sequence.

OPTIONS COMPATIBILITY

Many of the short and long forms of **pcre2grep**'s options are the same as in the GNU **grep** program. Any long option of the form **--xxx-regexp** (GNU terminology) is also available as **--xxx-regex** (PCRE2 terminology). However, the **--depth-limit**, **--file-list**, **--file-offsets**, **--heap-limit**, **--include-dir**, **--line-offsets**, **--locale**, **--match-limit**, **-M**, **--multiline**, **-N**, **--newline**, **--om-separator**, **--output**, **-u**, **--utf**, **-U**, and **--utf-allow-invalid** options are specific to **pcre2grep**, as is the use of the **--only-matching** option with a capturing parentheses number.

Although most of the common options work the same way, a few are different in **pcre2grep**. For example, the **--include** option's argument is a glob for GNU **grep**, but a regular expression for **pcre2grep**. If both the **-c** and **-l** options are given, GNU grep lists only file names, without counts, but **pcre2grep** gives the counts

as well.

OPTIONS WITH DATA

There are four different ways in which an option with data can be specified. If a short form option is used, the data may follow immediately, or (with one exception) in the next command line item. For example:

```
-f/some/file
-f /some/file
```

The exception is the **-o** option, which may appear with or without data. Because of this, if data is present, it must follow immediately in the same item, for example -o3.

If a long form option is used, the data may appear in the same command line item, separated by an equals character, or (with two exceptions) it may appear in the next command line item. For example:

```
--file=/some/file
--file /some/file
```

Note, however, that if you want to supply a file name beginning with ~ as data in a shell command, and have the shell expand ~ to a home directory, you must separate the file name from the option, because the shell does not treat ~ specially unless it is at the start of an item.

The exceptions to the above are the **--colour** (or **--color**) and **--only-matching** options, for which the data is optional. If one of these options does have data, it must be given in the first form, using an equals character. Otherwise **pcre2grep** will assume that it has no data.

USING PCRE2'S CALLOUT FACILITY

pcre2grep has, by default, support for calling external programs or scripts or echoing specific strings during matching by making use of PCRE2's callout facility. However, this support can be completely or partially disabled when **pcre2grep** is built. You can find out whether your binary has support for callouts by running it with the **--help** option. If callout support is completely disabled, all callouts in patterns are ignored by **pcre2grep**. If the facility is partially disabled, calling external programs is not supported, and callouts that request it are ignored.

A callout in a PCRE2 pattern is of the form (?C<arg>) where the argument is either a number or a quoted string (see the **pcre2callout** documentation for details). Numbered callouts are ignored by **pcre2grep**; only callouts with string arguments are useful.

Calling external programs or scripts

This facility can be independently disabled when **pcre2grep** is built. It is supported for Windows, where a call to **_spawnvp()** is used, for VMS, where **lib$spawn()** is used, and for any other Unix-like environment where **fork()** and **execv()** are available.

If the callout string does not start with a pipe (vertical bar) character, it is parsed into a list of substrings separated by pipe characters. The first substring must be an executable name, with the following substrings specifying arguments:

```
executable_name|arg1|arg2|...
```

Any substring (including the executable name) may contain escape sequences started by a dollar character: $<digits> or ${<digits>} is replaced by the captured substring of the given decimal number, which must be greater than zero. If the number is greater than the number of capturing substrings, or if the capture is unset, the replacement is empty.

Any other character is substituted by itself. In particular, $$ is replaced by a single dollar and $| is replaced by a pipe character. Here is an example:

```
echo -e "abcde\n12345" | pcre2grep \
  '(?x)(.)(..(.))
  (?C"/bin/echo|Arg1: [$1] [$2] [$3]|Arg2: $|${1}$| ($4)")()' -
```

Output:

```
Arg1: [a] [bcd] [d] Arg2: |a| ()
abcde
Arg1: [1] [234] [4] Arg2: |1| ()
12345
```

The parameters for the system call that is used to run the program or script are zero-terminated strings. This means that binary zero characters in the callout argument will cause premature termination of their substrings, and therefore should not be present. Any syntax errors in the string (for example, a dollar not followed by another character) cause the callout to be ignored. If running the program fails for any reason (including the non-existence of the executable), a local matching failure occurs and the matcher backtracks in the normal way.

Echoing a specific string

This facility is always available, provided that callouts were not completely disabled when **pcre2grep** was built. If the callout string starts with a pipe (vertical bar) character, the rest of the string is written to the output, having been passed through the same escape processing as text from the --output option. This provides a simple echoing facility that avoids calling an external program or script. No terminator is added to the string, so if you want a newline, you must include it explicitly. Matching continues normally after the string is output. If you want to see only the callout output but not any output from an actual match, you should end the relevant pattern with (*FAIL).

MATCHING ERRORS

It is possible to supply a regular expression that takes a very long time to fail to match certain lines. Such patterns normally involve nested indefinite repeats, for example: (a+)*\d when matched against a line of a's with no final digit. The PCRE2 matching function has a resource limit that causes it to abort in these circumstances. If this happens, **pcre2grep** outputs an error message and the line that caused the problem to the standard error stream. If there are more than 20 such errors, **pcre2grep** gives up.

The **--match-limit** option of **pcre2grep** can be used to set the overall resource limit. There are also other limits that affect the amount of memory used during matching; see the discussion of **--heap-limit** and **--depth-limit** above.

DIAGNOSTICS

Exit status is 0 if any matches were found, 1 if no matches were found, and 2 for syntax errors, overlong lines, non-existent or inaccessible files (even if matches were found in other files) or too many matching errors. Using the **-s** option to suppress error messages about inaccessible files does not affect the return code.

When run under VMS, the return code is placed in the symbol PCRE2GREP_RC because VMS does not distinguish between exit(0) and exit(1).

SEE ALSO

pcre2pattern(3), **pcre2syntax**(3), **pcre2callout**(3).

AUTHOR

Philip Hazel
University Computing Service
Cambridge, England.

REVISION

Last updated: 15 June 2019