grep - Unix, Linux Command

NAME

grep - 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 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.

OPTIONS

Tag	Description
-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 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.

--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 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.
-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 newlines, 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 contains 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 coming 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 immediately 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 equivalent 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 conform 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-Windows.
-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 version 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 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.
-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.

EXAMPLES

Example-1:

To Search for the given string in a single file test.sh

```
$ cat test.sh
#!/bin/bash
fun()
        echo "This is a test."
        # Terminate our shell script with success message
```

```
        exit 1

fun()
```

from above file grep exit:

```
$ grep "exit" demo_file
```

output:

```
        exit 1
```

Example-2:

To Checking for the given string in multiple files: in this case test.sh and test1.sh

```
$ cat test.sh
#!/bin/bash
fun()
        echo "This is a test."
        # Terminate our shell script with success message
        exit 1

fun()
```

```
$ cat test1.sh
#!/bin/bash
fun()
        echo "This is a test1."
        # Terminate our shell script with success message
        exit 0

fun()
```

grep exit in both files test.sh and test1.sh:

```
$ grep exit test*
```

output:

```
test1.sh:       exit 0
test.sh:        exit 1
```

Example-3:

To Case insensitive search using grep -i, added EXIT in test1.sh

```
$ cat test1.sh
#!/bin/bash
fun()
        echo "This is a test1."
        # Terminate our shell script with success message, EXIT with 0
        exit 0

fun()
```

```
$ grep exit test1.sh
```

test1.sh:	exit 0

$ grep -i exit test*

output:

test1.sh:	# Terminate our shell script with success message, EXIT with 0
test1.sh:	exit 0

two lines with -i option, as its case insensitive.

output:

	echo "This is a test1."

Example-5:

* To Checking for full words, not for sub-strings using grep -w

$ grep -w "exit" test.sh

output:

	exit 1

Note:

$ grep -w ex test.sh -> no output, as full word with ex is not in test.sh

for substring search we should not use -w option.

$ grep ex test.sh

	exit 1

Example-6:

To Invert match using grep -v

$ cat test.sh
#!/bin/bash
fun()
	echo "This is a test."
	# Terminate our shell script with success message
	exit 1

fun()

$ grep -v exit test.sh

output:

#!/bin/bash
fun()
	echo "This is a test."
	# Terminate our shell script with success message

Note: exit statement is not seen in output.

Example-7:

To print line number of searched string

$ grep -n exit test.sh

output:

5: exit 1