



[nixCraft](#) → [Howto](#) → [BASH Shell](#) → Regular expressions in grep (regex) with examples



Regular expressions in grep (regex) with examples

Author: Vivek Gite • Last updated: June 17, 2023 • [82 comments](#)

How do I use the [grep command](#) with regular expressions on a Linux and Unix-like operating systems? How do I use grep and regular expressions (regex) to search for text/ words in Linux?



Linux comes with GNU grep, which supports extended regular expressions. GNU grep is the default on all Linux systems. The [grep command](#) is used to locate information stored anywhere on your server or workstation. Let us see fundamental of regex and how to use regular expressions in the Linux and Unix like systems.

Tutorial details	
Difficulty level	Easy
Root privileges	No
Requirements	Linux or Unix terminal
Category	Searching
Prerequisites	egrep command
OS compatibility	BSD • Linux • macOS • Unix • WSL
Est. reading time	8 minutes

nixCraft: Privacy First, Reader Supported

→ **nixCraft is a one-person operation.** I create all the content myself, with no help from AI or ML. I keep the content accurate and up-to-date.

→ **Your privacy is my top priority.** I don't track you, show you ads, or spam you with emails. Just pure content in the true spirit of Linux and FLOSS.

→ **Fast and clean browsing experience.** nixCraft is designed to be fast and easy to use. You won't have to deal with pop-ups, ads, cookie banners, or other distractions.

→ **Support independent content creators.** nixCraft is a labor of love, and it's only possible thanks to the support of our readers. If you enjoy the content, please support us on Patreon or share this page on social media or your blog. Every bit helps.

[Join Patreon →](#)

Regular Expressions in grep

Regular Expressions is nothing but a pattern to match for each input line. A pattern is a sequence of characters. Following all are examples of pattern:

```
^w1
w1|w2
[ ^ ]
foo
bar
[0-9]
```

Three types of regex

The grep understands three different types of regular expression syntax as follows:

1. basic (BRE)
2. extended (ERE)
3. perl (PCRE)

grep Regular Expressions Examples

Search for a word named 'vivek' in the [/etc/passwd](#) file:

```
$ grep 'vivek' /etc/passwd
```

Sample outputs:

```
vivek:x:1000:1000:Vivek Gite,,,:/home/vivek:/bin/bash
vivekgite:x:1001:1001::/home/vivekgite:/bin/sh
gitevivek:x:1002:1002::/home/gitevivek:/bin/sh
```

Next, search for a word named 'vivek' in any case (i.e. case insensitive search):

```
$ grep -i -w 'vivek' /etc/passwd
```

Let us try to search two words 'vivek' or 'raj' in any case:

```
$ grep -E -i -w 'vivek|raj' /etc/passwd
```

The PATTERN in last example, used as an extended regular expression. The following will match word Linux or UNIX in any case using the [egrep command](#):

```
$ egrep -i '^(linux|unix)' filename
# Same as above by passing the '-E' to the grep #
$ grep -E -i '^(linux|unix)' filename
```

A note about `egrep` vs. `grep -E` syntax

The latest version of egrep will show the following warning

```
egrep: warning: egrep is obsolescent; using grep -E
Usage: grep [OPTION]... PATTERNS [FILE]...
Try 'grep --help' for more information.
```

You need to update all your scripts and command to use the following syntax. From:

```
$ egrep -i 'foo|bar' /path/to/file
```

To (avoid using the `egrep`):

```
$ grep -E -i 'foo|bar' /path/to/file
```

How to match single characters

The `.` character (period, or dot) matches any one character. Consider the following demo.txt file:

```
$ cat demo.txt
```

Sample outputs:

```
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
```

```
purchase1.db  
purchase2.db  
purchase3.db  
purchase.idx  
foo2.txt  
bar.txt
```

Let us find all filenames starting with purchase, type:

```
$ grep 'purchase' demo.txt
```

Next I need to find all filenames starting with purchase and followed by another character:

```
$ grep 'purchase.db' demo.txt
```

Our final example find all filenames starting with purchase but ending with db:

```
$ grep 'purchase..db' demo.txt
```

```
vivek@nixcraft-asus:/tmp$ cat demo.txt
List of files:
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
bar.txt
vivek@nixcraft-asus:/tmp$ grep 'purchase' demo.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
vivek@nixcraft-asus:/tmp$ grep 'purchase.' demo.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
vivek@nixcraft-asus:/tmp$ grep 'purchase.db' demo.txt
purchase.db
vivek@nixcraft-asus:/tmp$ grep 'purchase..db' demo.txt
purchase1.db
purchase2.db
purchase3.db
vivek@nixcraft-asus:/tmp$
```

© www.cyberciti.biz

How to match only dot (.)

A dot (.) has a special meaning in regex, i.e. match any character. But, what if you need to match dot (.) only? I want to tell my grep command that I want actual dot (.) character and not the regex special meaning of the . (dot) character. You can escape the dot (.) by preceding it with a \ (backslash):

```
$ grep 'purchase..' demo.txt
$ grep 'purchase.\.' demo.txt
```

```
vivek@nixcraft-asus:/tmp$ grep 'purchase..' demo.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
vivek@nixcraft-asus:/tmp$ grep 'purchase\.' demo.txt
purchase1.db
purchase2.db
purchase3.db
vivek@nixcraft-asus:/tmp$
```

© www.cyberciti.biz

Anchors

You can use ^ and \$ to force a regex to match only at the start or end of a line, respectively. The following example displays lines starting with the vivek only:

```
$ grep ^vivek /etc/passwd
```

Sample outputs:

```
vivek:x:1000:1000:Vivek Gite,,,:/home/vivek:/bin/bash
vivekgite:x:1001:1001::/home/vivekgite:/bin/sh
```

You can display only lines starting with the word vivek only i.e. do not display vivekgite, vivekg etc:

```
$ grep -w ^vivek /etc/passwd
```

Find lines ending with word foo:

```
$ grep 'foo$' filename
```

Match line only containing foo:

```
$ grep '^foo$' filename
```

You can search for blank lines with the following examples:

```
$ grep '^$' filename
```

Matching Sets of Characters

How to match sets of character using grep

The dot (.) matches any single character. You can match specific characters and character ranges using [.] syntax. Say you want to Match both 'Vivek' or 'vivek':

```
$ grep '[vV]ivek' filename
```

OR

```
$ grep '[vV][iI][Vv][Ee][kK]' filename
```

Let us match digits and upper and lower case characters. For example, try to match words such as vivek1, Vivek2 and so on:

```
$ grep -w '[vV]ivek[0-9]' filename
```

In this example match two numeric digits. In other words match foo11, foo12, foo22 and so on, enter:

```
$ grep 'foo[0-9][0-9]' filename
```


You are not limited to digits, you can match at least one letter:

```
$ grep '[A-Za-z]' filename
```

Display all the lines containing either a “w” or “n” character:

```
$ grep [wn] filename
```

Within a bracket expression, the name of a character class enclosed in “[:” and “:]” stands for the list of all characters belonging to that class. Standard character class names are:

- `[[:alnum:]]` – Alphanumeric characters.
- `[[:alpha:]]` – Alphabetic characters
- `[[:blank:]]` – Blank characters: space and tab.
- `[[:digit:]]` – Digits: ‘0 1 2 3 4 5 6 7 8 9’.
- `[[:lower:]]` – Lower-case letters: ‘a b c d e f g h i j k l m n o p q r s t u v w x y z’.
- `[[:space:]]` – Space characters: tab, newline, vertical tab, form feed, carriage return, and space.
- `[[:upper:]]` – Upper-case letters: ‘A B C D E F G H I J K L M N O P Q R S T U V W X Y Z’.


In this example match all upper case letters:

```
$ grep '[:upper:]' filename
```

How negates matching in sets

The ^ negates all ranges in a set:

```
$ grep '[vV]ivek[^0-9]' test
```

A terminal window with a dark purple background. The prompt is 'vivek@nixcraft-asus:/tmp\$'. The user runs 'cat test', showing a file with several lines of text. Then, the user runs 'grep '[vV]ivek' test', and the output shows the same text with 'vivek' and 'Vivek' highlighted in red. Finally, the user runs 'grep '[vV]ivek[^0-9]' test', and the output shows the same text with 'vivek' and 'Vivek' highlighted in red, but the trailing spaces and newlines are not. The terminal ends with a blank line and a cursor.

```
vivek@nixcraft-asus:/tmp$ cat test
My name it Vivek Gite.
vivek loves Unix and Linux too.
My Linux user name is vivek and password is *****
My CentOS user name is vivek1 and password is *****
vivek@nixcraft-asus:/tmp$ grep '[vV]ivek' test
My name it Vivek Gite.
vivek loves Unix and Linux too.
My Linux user name is vivek and password is *****
My CentOS user name is vivek1 and password is *****
vivek@nixcraft-asus:/tmp$ grep '[vV]ivek[^0-9]' test
My name it Vivek Gite.
vivek loves Unix and Linux too.
My Linux user name is vivek and password is *****
vivek@nixcraft-asus:/tmp$
```

© www.cyberciti.biz

Using grep regular expressions to search for text patterns

Wildcards

You can use the “.” for a single character match. In this example match all 3 character word starting with “b” and ending in “t”:

```
grep "\<b.t\>" filename
```

Where,

- \< Match the empty string at the beginning of word
- \> Match the empty string at the end of word.

Print all lines with exactly two characters:

```
$ grep '^..$' filename
```

Display any lines starting with a dot and digit:

```
$ grep '^\. [0-9]' filename
```

Escaping the dot

Say you just want to match an IP address 192.168.2.254 and nothing else. The following regex to find an IP address 192.168.1.254 will not work (remember the dot matches any single character?):

```
$ grep '192.168.1.254' hosts
```

Sample outputs:

```
192.168.2.18      centos7
192x168y2z18      centos7
```

All three dots need to be escaped:

```
192.168.2.18      centos7
```

```
$ grep '192\.168\.1\.254' hosts
```

The following example will only match an IP address:

```
$ grep -E '[[[:digit:]]{1,3}\. [[[:digit:]]{1,3}\. [[[:digit:]]{1,3}\.
[[[:digit:]]{1,3}' file
```

How Do I Search a Pattern Which Has a Leading Symbol?

Searches for all lines matching '`--test--`' using `-e` option Without `-e`, grep would attempt to parse '`-test-`' as a list of options. For example:

```
$ grep -e '--test--' filename
```

How Do I do OR with grep?

Use the following syntax:

```
$ grep -E 'word1|word2' filename
### OR ###
$ egrep 'word1|word2' filename
```

OR Try the following syntax:

```
$ grep 'word1\|word2' filename
```

How do I AND with grep?

Use the following syntax to display all lines that contain both 'word1' and 'word2'

```
$ grep 'word1' filename | grep 'word2'
```

OR try the following syntax:

```
$ grep 'foo.*bar\|word3.*word4' filename
```

How Do I Test Sequence?

You can test how often a character must be repeated in sequence using the following syntax:

`{N}`

`{N,}`

`{min,max}`

Match a character “v” two times:

```
$ egrep "v{2}" filename
```

The following will match both “col” and “cool” words:

```
$ egrep 'co{1,2}l' filename
```

Our next example will match any row of at least three letters ‘c’.

```
$ egrep 'c{3,}' filename
```

In this example, I will match mobile number which is in the following format 91-1234567890 (i.e TwoDigit-TenDigit)

```
$ grep "[[:digit:]]\{2\}[ -]\?[[[:digit:]]\{10\}" filename
```

How Do I Highlight with grep?

Pass the `--color` as follows:

```
$ grep --color regex filename
```

How Do I Show Only The Matches, Not The Lines?

Use the following syntax:

```
$ grep -o regex filename
```

grep Regular Expression Operator

I hope following table will help you quickly understand regular expressions in grep when using under Linux or Unix-like systems:

Table 1: grep (egrep) regex operator		
Operator	Description	Exam
<div>.</div>	Matches any single character.	<div>grep '.' file</div> <div>grep 'foo.' input</div>
<div>?</div>	The preceding item is optional and will be matched , at most, once.	<div>grep 'vivek?' /etc</div>
<div>*</div>	The preceding item will be matched zero or more times.	<div>grep 'vivek*' /etc</div>
<div>+</div>	The preceding item will be matched one or more times.	<div>ls /var/log/ grep</div>
<div>{N}</div>	The preceding item is matched exactly N times.	<div>egrep '[0-9]{2} in</div>
<div>{N, }</div>	The preceding item is matched N or more times.	<div>egrep '[0-9]{2,} ir</div>
<div>{N, M}</div>	The preceding item is matched at least N times, but not more than M times.	<div>egrep '[0-9]{2,4}</div>

Operator	Description	Example
-	Represents the range if it's not first or last in a list or the ending point of a range in a list.	<code>grep ':/bin/[a-z]*'</code>
^	Matches the empty string at the beginning of a line; also represents the characters not in the range of a list.	<code>grep '^vivek' /etc</code> <code>grep '[^0-9]*' /etc</code>
\$	Matches the empty string at the end of a line.	<code>grep '^\$' /etc/passwd</code>
\b	Matches the empty string at the edge of a word.	<code>grep '\bvivek' /etc</code>
\B	Matches the empty string provided it's not at the edge of a word.	<code>grep '\B/bin/bash' /etc</code>
\<	Match the empty string at the beginning of word.	<code>grep '\<vivek' /etc</code>
\>	Match the empty string at the end of word.	<code>grep 'bash\>' /etc</code> <code>grep '\<vivek\>' /etc</code>

Linux grep vs egrep command

The egrep is the same as `grep -E` command. It interprets PATTERN as an extended regular expression. From the grep man page:

In basic regular expressions the meta-characters `?`, `+`, `{`, `|`, `(`, `\|`, `\(`, and `\)`.

Traditional egrep did not support the `{` meta-character, and so grep -E patterns and should use `[{]}` to match a literal `{`.

GNU grep -E attempts to support traditional usage by assuming
For example, the command grep -E '{1}' searches for the two-cha
POSIX.2 allows this behavior as an extension, but portable scr

Conclusion

You learned how to regular expressions (regex) in grep running on Linux or Unix with various examples. See GNU/grep man page online here or see the following resources using the [man command](#) or info command (or pass the [--help option](#):

```
$ man egrep
$ info egrep
$ man 7 regex
$ egrep --help
```

This entry is **2 of 7** in the **Linux / UNIX grep Command Tutorial** series. Keep reading the rest of the series:

1. [How To Use grep Command In Linux / UNIX](#)
2. Regular Expressions In grep
3. [Search Multiple Words / String Pattern Using grep Command](#)
4. [Grep Count Lines If a String / Word Matches](#)
5. [Grep From Files and Display the File Name](#)
6. [How To Find Files by Content Under UNIX](#)
7. [grep command: View Only Configuration File Directives](#)

Did you notice? 🤔