# options

-i: ignore the case of your search term

-v: show lines that don’t match, instead of those that do

-c: instead of returning matches, return the number of matches

-x: return only an exact match

-E: interpret search as an extended regular expression

-F: interpret search as a list of fixed strings, including newlines, dots, etc

-f: get the search patterns from this file

-H: print the filename with each match

-m: stop reading file after n number of matches

-n: print the line number of where matches were found

-q: don’t output anything, but exit with status 0 if any match is found (check that status with echo $?).

-A: print n number of lines after the match

-B: print n number of lines before the match

-o: print only the matching part of the line

-e: search literally, and protects patterns starting with a hyphen

-w: find matches surrounded by space

--color: add color to the matched output

--help: get some help

-V: get grep’s version

# using regular expressions

grep can also use regular expressions. Here are some of the most common ones to know about:

[ NOTE: In GNU grep there is no difference between basic and extended regular expressions, and the functionality of egrep and fgrep have been pulled into grep itself, so there’s no reason to use them anymore. ]

Regex structure

These are the basic building blocks of a regex.

: disregard the system meaning of (escape) the next character. Useful when entering carats for code, dots for IP addresses, etc.

[ ]: a bracket is a list of characters, and matches any character in that list. If the first character is ^ then it matches what’s not in the list

-: a hyphen indicates a range, so [a-d] means [abcd]

[^ ]: shows what doesn’t include those characters

^: matches at the beginning of the line

$: matches at the end of the line

.: matches any single character, except end of line

+: matches one or more of the preceding thing (at least once)

\*: matches zero or more of the preceding thing

{x,y}: matches x to y occurrences of the preceding thing

{x}: matches exactly x occurrences of the preceding thing

{x,}: matches x or more occurrences of the preceding thing

# Predefined shortcuts

A number of common expressions have been defined as universal shortcuts to make your searches easier, and they use double brackets.

[[:alnum:]]: any alphanumeric character

[[:alpha:]]: any alphabetic character

[[:contrl:]]: any control character

[[:digit:]]: any number

[[:lower:]]: any lower case character

[[:print:]]: any printable character

[[:space:]]: any space character, including space, tab, newline, CR, FF, etc.

# Examples

So if you have a file called names.txt:

cat names.txt

Sarah

John

Michael

Stewart

Christina

## ignoring case

One of the most common mistakes is searching for something and not getting a hit because there was a case mismatch. You can simply ignore case with the -i option:

grep -i jill file.txt

Jill

## getting non-matches

Another extremely common situation is wanting to get every line that doesn’t have the search pattern in it. You can do that with the -v option:

grep -v Christina file.txt

Stewart

- 441

<a href="https://www.google.com">Google</a>

Jill

54r4h

Shazbot123

lll

221

Item 1, Item 2, Item 3

TABS TABS TABS

## searching in multiple files

It’s easy to search in multiple files. Simply include the capture of each file in your file target.

grep -i password file\*.txt

[ NOTE: The file\*.txt will match file1.txt, file2, text, etc., as expected. ]

So you could do something like this as well:

grep -i password \*.txt

In this case, the \*.txt would search within all text files in the current directory.

## find lines with numbers in them

Here we’re going to switch to a shortcut:

grep '[[:digit:]]' names.txt

- 441

54r4h

Shazbot123

lll

221

Item 1, Item 2, Item 3

## find lines that start with numbers

Here we’re going to add the ^ to look for all lines that start with a number.

grep '^[[:digit:]]' names.txt

54r4h

221

## exact matches

Sometimes you want to find a line that is exactly what you searched for, rather than lines that have it in it.

echo "Jason" | grep -x son

## find matches surrounded by space

grep -w 1 file.txt

Item 1, Item 2, Item 3

[ NOTE: This matched because of the final number, the 3. ]

## find ip addresses

grep '[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}' net.txt

23.44.124.67

172.16.23.1

10.10.2.21

## show lines around the match

Here’s the -A option, which gets lines after the match.

grep -A 2 Three count.txt

Three

Four

Five

[ NOTE: -B shows the lines before, and -C shows the lines both before and after. ]