In this article let us review 15 practical examples of Linux grep command that will be very useful to both newbies and experts.

First create the following demo_file that will be used in the examples below to demonstrate grep command.

```
$ cat demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.

this line is the 1st lower case line in this file.

This Line Has All Its First Character Of The Word With Upper Case.



Two lines above this line is empty.

And this is the last line.
```

1. Search for the given string in a single file

The basic usage of grep command is to search for a specific string in the specified file as shown below.

```
Syntax:

grep "literal_string" filename

$ grep "this" demo_file

this line is the 1st lower case line in this file.

Two lines above this line is empty.

And this is the last line.
```

2. Checking for the given string in multiple files.

```
Syntax:
```

```
grep "string" FILE_PATTERN
```

This is also a basic usage of grep command. For this example, let us copy the demo_file to demo_file1. The grep output will also include the file name in front of the line that matched the specific pattern as shown below. When the Linux shell sees the meta character, it does the expansion and gives all the files as input to grep.

```
$ cp demo_file demo_file1



$ grep "this" demo_*

demo_file:this line is the 1st lower case line in this file.

demo_file:Two lines above this line is empty.

demo_file:And this is the last line.

demo_file1:this line is the 1st lower case line in this file.

demo_file1:Two lines above this line is empty.

demo_file1:And this is the last line.
```

3. Case insensitive search using grep -i

```
Syntax:

grep -i "string" FILE
```

This is also a basic usage of the grep. This searches for the given string/pattern case insensitively. So it matches all the words such as “the”, “THE” and “The” case insensitively as shown below.

```
$ grep -i "the" demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.

this line is the 1st lower case line in this file.

This Line Has All Its First Character Of The Word With Upper Case.
```

```
And this is the last line.
```

4. Match regular expression in files

```
Syntax:

grep "REGEX" filename
```

This is a very powerful feature, if you can use use regular expression effectively. In the following example, it searches for all the pattern that starts with “lines” and ends with “empty” with anything in-between. i.e To search “lines[anything in-between]empty” in the demo_file.

```
$ grep "lines.*empty" demo_file

Two lines above this line is empty.
```

From documentation of grep: A regular expression may be followed by one of several repetition operators:

- ? 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.
- {,m} The preceding item is matched at most m times.
- {n,m} The preceding item is matched at least n times, but not more than m times.

5. Checking for full words, not for sub-strings using grep -w

If you want to search for a word, and to avoid it to match the substrings use -w option. Just doing out a normal search will show out all the lines.

The following example is the regular grep where it is searching for “is”. When you search for “is”, without any option it will show out “is”, “his”, “this” and everything which has the substring “is”.

```
$ grep -i "is" demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.

this line is the 1st lower case line in this file.
```

```
This Line Has All Its First Character Of The Word With Upper Case.

Two lines above this line is empty.

And this is the last line.
```

The following example is the WORD grep where it is searching only for the word “is”. Please note that this output does not contain the line “This Line Has All Its First Character Of The Word With Upper Case”, even though “is” is there in the “This”, as the following is looking only for the word “is” and not for “this”.

```
$ grep -iw "is" demo_file

THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.

this line is the 1st lower case line in this file.

Two lines above this line is empty.

And this is the last line.
```

6. Displaying lines before/after/around the match using grep -A, -B and -C

When doing a grep on a huge file, it may be useful to see some lines after the match. You might feel handy if grep can show you not only the matching lines but also the lines after/before/around the match.

Please create the following demo_text file for this example.

```
$ cat demo_text

4. Vim Word Navigation



You may want to do several navigation in relation to the words, such as:



 * e - go to the end of the current word.

 * E - go to the end of the current WORD.
```

```
 * b - go to the previous (before) word.

 * B - go to the previous (before) WORD.

 * w - go to the next word.

 * W - go to the next WORD.



WORD - WORD consists of a sequence of non-blank characters, separated with white
space.

word - word consists of a sequence of letters, digits and underscores.



Example to show the difference between WORD and word



 * 192.168.1.1 - single WORD

 * 192.168.1.1 - seven words.
```

6.1 Display N lines after match
-A is the option which prints the specified N lines after the match as shown below.

```
Syntax:

grep -A <N> "string" FILENAME
```

The following example prints the matched line, along with the 3 lines after it.

```
$ grep -A 3 -i "example" demo_text

Example to show the difference between WORD and word



* 192.168.1.1 - single WORD
```

```
* 192.168.1.1 - seven words.
```

6.2 Display N lines before match
-B is the option which prints the specified N lines before the match.

```
Syntax:

grep -B <N> "string" FILENAME
```

When you had option to show the N lines after match, you have the -B option for the opposite.

```
$ grep -B 2 "single WORD" demo_text

Example to show the difference between WORD and word



* 192.168.1.1 - single WORD
```

6.3 Display N lines around match
-C is the option which prints the specified N lines before the match. In some occasion you might want the match to be appeared with the lines from both the side. This options shows N lines in both the side(before & after) of match.

```
$ grep -C 2 "Example" demo_text

word - word consists of a sequence of letters, digits and underscores.



Example to show the difference between WORD and word



* 192.168.1.1 - single WORD
```

7. Highlighting the search using GREP_OPTIONS

As grep prints out lines from the file by the pattern / string you had given, if you wanted it to highlight which part matches the line, then you need to follow the

following way.

When you do the following export you will get the highlighting of the matched searches. In the following example, it will highlight all the this when you set the GREP_OPTIONS environment variable as shown below.

```
$ export GREP_OPTIONS='--color=auto' GREP_COLOR='100;8'
```

```
$ grep this demo_file

this line is the 1st lower case line in this file.
Two lines above this line is empty.
And this is the last line.
```

8. Searching in all files recursively using grep -r

When you want to search in all the files under the current directory and its sub directory. -r option is the one which you need to use. The following example will look for the string “ramesh” in all the files in the current directory and all it’s subdirectory.

```
$ grep -r "ramesh" *
```

9. Invert match using grep -v

You had different options to show the lines matched, to show the lines before match, and to show the lines after match, and to highlight match. So definitely You’d also want the option -v to do invert match.

When you want to display the lines which does not matches the given string/pattern, use the option -v as shown below. This example will display all the lines that did not match the word “go”.

```
$ grep -v "go" demo_text

4. Vim Word Navigation



You may want to do several navigation in relation to the words, such as:
```

```
WORD - WORD consists of a sequence of non-blank characters, separated with white
space.

word - word consists of a sequence of letters, digits and underscores.



Example to show the difference between WORD and word



* 192.168.1.1 - single WORD

* 192.168.1.1 - seven words.
```

10. display the lines which does not matches all the given pattern.

```
Syntax:

grep -v -e "pattern" -e "pattern"

$ cat test-file.txt

a

b

c

d



$ grep -v -e "a" -e "b" -e "c" test-file.txt

d
```

11. Counting the number of matches using grep -c

When you want to count that how many lines matches the given pattern/string, then use the option -c.

```
Syntax:

grep -c "pattern" filename

$ grep -c "go" demo_text

6
```

When you want do find out how many lines matches the pattern

```
$ grep -c this demo_file

3
```

When you want do find out how many lines that does not match the pattern

```
$ grep -v -c this demo_file

4
```

12. Display only the file names which matches the given pattern using grep -l

If you want the grep to show out only the file names which matched the given pattern, use the -l (lower-case L) option.

When you give multiple files to the grep as input, it displays the names of file which contains the text that matches the pattern, will be very handy when you try to find some notes in your whole directory structure.

```
$ grep -l this demo_*

demo_file

demo_file1
```

13. Show only the matched string

By default grep will show the line which matches the given pattern/string, but if you want the grep to show out only the matched string of the pattern then use the -o option.

It might not be that much useful when you give the string straight forward. But it becomes very useful when you give a regex pattern and trying to see what it matches as

```
$ grep -o "is.*line" demo_file

is line is the 1st lower case line

is line

is is the last line
```

14. Show the position of match in the line

When you want grep to show the position where it matches the pattern in the file, use the following options as

```
Syntax:

grep -o -b "pattern" file

$ cat temp-file.txt

12345

12345



$ grep -o -b "3" temp-file.txt

2:3

8:3
```

Note: The output of the grep command above is not the position in the line, it is byte offset of the whole file.

15. Show line number while displaying the output using grep -n

To show the line number of file with the line matched. It does 1-based line numbering for each file. Use -n option to utilize this feature.

```
$ grep -n "go" demo_text

5: * e - go to the end of the current word.

6: * E - go to the end of the current WORD.

7: * b - go to the previous (before) word.

8: * B - go to the previous (before) WORD.

9: * w - go to the next word.

10: * W - go to the next WORD.
```