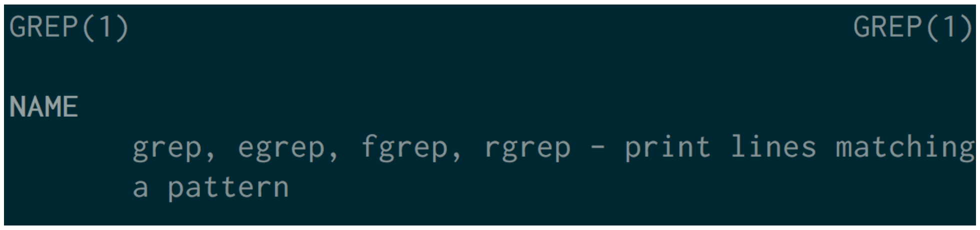
Last updated Saturday, Nov 16, 2019

Linux and Unix grep command tutorial with examples

Tutorial using grep, a UNIX and Linux command to print lines matching a pattern. Examples of finding text in a file, printing line numbers, counting the number of matches, searching recursively and ignoring case sensitivity.

Estimated reading time: 3 minutes

Table of contents



```
GREP(1)                                              GREP(1)

NAME
       grep, egrep, fgrep, rgrep - print lines matching
       a pattern
```

What is the grep command in UNIX?

The `grep` command in UNIX is a command line utility for printing lines that match a pattern. It can be used to find text in a file and search a directory structure of files recursively. It also supports showing the context of a match by showing lines before and after the result and has support for regular expressions in pattern matching.

How to find text in a file

To find text in a file pass the string you are looking for to `grep` followed by the name of the file or files.

```
grep 'computer' /usr/share/dict/words
computer
```

The `grep` tool will print occurrences that it finds to standard output.

How to list line numbers for matches

To list line numbers and file names pass the `-n` option to grep. This prints matches to standard output along with the line number it was found on.

```
grep 'computer' -n /usr/share/dict/words
40565
```

This can be useful if you are looking to edit a file and want to launch vim and go straight to the line.

```
vim +40565 /usr/share/dict/words
```

How to print lines before and after a match

To print lines before and after a match the `-A` and `-B` options can be used. Both expect a number and will print this number of lines.

```
grep -B 2 -A 2 'computer' /usr/share/dict/words
computativeness
compute
computer
computist
computus
```

Using the `-A` and `-B` options can be very useful for grepping through log files to see what occurred before and after the item of interest.

A further option is available in `-C` that will print the context of the match. This is equivalent to using both `-A` and `-B` .

```
grep -C 2 'computer' /usr/share/dict/words
computativeness
compute
computer
computist
computus
```

The `--context` option may also be used and defaults to two lines before and after if no number is given.

How to count the number of matches

To count the number of matches use the `-c` option. This outputs a number count to standard output.

```
grep -c 'comput*' /usr/share/dict/words
50
```

How to print the filename for a match

To print the filename for a match use the `-H` option. This is automatically invoked when `grep` is given more than one file to search.

```
grep -H 'computer' /usr/share/dict/words
/usr/share/dict/words:computer
```

How to search recursively

To search for a pattern recursively use the `-R` option. This will search through all files in the directory tree that you have permission to read.

```
grep -R 'passwd' /etc
/etc/pam.d/su:# NIS (man nsswitch) as well as normal /etc/passwd and
/etc/pam.d/chpasswd:# The PAM configuration file for the Shadow 'chpa
```

How to search for the inverse of a pattern

To search for the inverse of a pattern use the `-v` option. This will print inverse matches to standard output.

```
grep -v 'computer' /usr/share/dict/words
A
a
aa
aal
....
```

How to ignore case when searching

To ignore case when searching use the `-i` option. By default `grep` will respect case.

```
grep 'COMPUTER' /usr/share/dict/words
# no match
grep -i 'COMPUTER' /usr/share/dict/words
computer
```

How to use basic regular expressions when searching

To use basic regular expressions all versions of `grep` support basic character matches. In the following example the pattern matches ‘ia’ characters at the end of the line.

```
grep 'ia$' /usr/share/dict/words
abasia
Abelia
abepithymia
....
```

A great book for understanding the power of regular expressions is Mastering Regular Expressions (http://shop.oreilly.com/product/9780596528126.do).

How to use extended regular expressions when searching

To use extended regular expressions use the `-e` option. The following line matches lines that do not contain the words ‘foo’ or ‘bar’.

```
grep -v -e 'foo' -e 'bar'
```

Note that in the GNU version of `grep` there is no difference in available functionality between basic and extended syntaxes.

Further reading