Unix/Linux grep command examples

By Alvin Alexander. Last updated: May 4, 2015
table of contents [show]

Linux *grep* commands FAQ: Can you share some Linux/Unix grep command examples?

Sure. The name *grep* means "general regular expression parser", but you can think of the `grep` command as a "search" command for Unix and Linux systems: It's used to search for text strings and more-complicated *regular expressions* within one or more files.

I think it's easiest to learn how to use the `grep` command by showing examples, so let's dive right in.

Abridged examples

First up, if you don't like reading a bunch of text and just want to see a collection of `grep` commands, this section is for you. (If the Table of Contents over there on the right side is still in the way, click or tap the 'hide' link in its title to hide it):

```
search for a string in one or more files
----------------------------------------
grep 'fred' /etc/passwd              # search for lines containing 'fred' in /etc/passwd
grep fred /etc/passwd                # quotes usually not when you don't use regex patterns
grep null *.scala                    # search multiple files

case-insensitive
----------------
grep -i joe users.txt                # find joe, Joe, JOe, JOE, etc.

regular expressions
-------------------
grep '^fred' /etc/passwd             # find 'fred', but only at the start of a line
grep '[FG]oo' *                      # find Foo or Goo in all files in the current dir
grep '[0-9][0-9][0-9]' *             # find all lines in all files in the current dir with three numbers in a row

display matching filenames, not lines
-------------------------------------
grep -l StartInterval *.plist        # show all filenames containing the string 'StartInterval'
grep -il StartInterval *.plist       # same thing, case-insensitive
```

```
show matching line numbers
--------------------------
grep -n we gettysburg-address.txt    # show line numbers as well as the matching lines

lines before and after grep match
---------------------------------
grep -B5 "the living" gettysburg-address.txt        # show all matches, and five lines before each match
grep -A10 "the living" gettysburg-address.txt       # show all matches, and ten lines after each match
grep -B5 -A5 "the living" gettysburg-address.txt    # five lines before and ten lines after

reverse the meaning
-------------------
grep -v fred /etc/passwd             # find any line *not* containing 'fred'
grep -vi fred /etc/passwd            # same thing, case-insensitive

grep in a pipeline
------------------
ps auxwww | grep httpd               # all processes containing 'httpd'
ps auxwww | grep -i java             # all processes containing 'java', ignoring case
ls -al | grep '^d'                   # list all dirs in the current dir

search for multiple patterns
----------------------------
egrep 'apple|banana|orange' *                                                    # search for multiple patterns, all files
in current dir
egrep -i 'apple|banana|orange' *                                                 # same thing, case-insensitive
egrep 'score|nation|liberty|equal' gettysburg-address.txt                        # all lines matching multiple patterns
locate -i calendar | grep Users | egrep -vi 'twiki|gif|shtml|drupal-7|java|PNG'   # oh yeah
       (see http://alvinalexander.com/linux-unix/linux-egrep-multiple-regular-expres...)

multiple search strings, multiple filename patterns
---------------------------------------------------
grep -li "jtable" $(find . -name "*.java,v" -exec grep -li "prevayl" {} \;)      # find all files named "*.java,v"
containing both
                                                                                 # 'prevayl' and 'jtable'

grep + find
-----------
find . -type f -exec grep -il 'foo' {} \;     # print all filenames of files under current dir containing 'foo', case-
insensitive
```

```
recursive grep search
---------------------
grep -rl 'null' .                           # very similar to the previous find command; does a recursive search
grep -ril 'null' /home/al/sarah /var/www    # search multiple dirs
egrep -ril 'aja|alvin' .                    # multiple patterns, recursive
      (see http://alvinalexander.com/linux-unix/recursive-grep-r-searching-egrep-find)

grep gzip files
---------------
zgrep foo myfile.gz                         # all lines containing the pattern 'foo'
zgrep 'GET /blog' access_log.gz             # all lines containing 'GET /blog'
zgrep 'GET /blog' access_log.gz | more      # same thing, case-insensitive
```

That's the short version of the `grep` examples. The rest of this document describes many of these examples.

Searching for a text string in one file

This first `grep` command example searches for all occurrences of the text string 'fred' within the */etc/passwd* file. It will find and display all of the lines in this file that contain the text string `fred`, including lines that contain usernames like "fred", and also other strings like "alfred":

```
grep 'fred' /etc/passwd
```

In a simple example like this, the quotes around the string `fred` aren't necessary, but they are needed if you're searching for a string that contains spaces, and will also be needed when you get into using regular expressions (search patterns).

Searching for a string in multiple files

Our next `grep` command example searches for all occurrences of the text string `joe` within all files of the current directory:

```
grep 'joe' *
```

The '*' wildcard matches all files in the current directory, and the grep output from this command will show both (a) the matching filename and (b) all lines in all files that contain the string 'joe'.

As a quick note, instead of searching all file with the "*" wildcard, you can also use grep to search all files in the current directory that end in the file extension *.txt*, like this:

```
grep 'joe' *.txt
```

Case-insensitive file searching with the Unix grep command

To perform a case-insensitive search with the grep command, just add the `-i` option, like this:

```
grep -i score gettysburg-address.txt
```

This grep search example matches the string "score", whether it is uppercase (SCORE), lowercase (score), or any mix of the two (Score, SCore, etc.).

Reversing the meaning of a grep search

You can reverse the meaning of a Linux grep search with the `-v` option. For instance, to show all the lines of my */etc/passwd* file that *don't* contain the string `fred`, I'd issue this command:

```
grep -v fred /etc/passwd
```

Using grep in a Unix/Linux command pipeline

The grep command is often used in a Unix/Linux pipeline. For instance, to show all the Apache `httpd` processes running on my Linux system, I use the grep command in a pipeline with the `ps` command:

```
ps auxwww | grep httpd
```

This returns the following output:

```
root     17937  0.0  0.0  14760  6880 ?        Ss   Apr01   0:39 /usr/local/apache/bin/httpd -k start
nobody   21538  0.0  0.0  24372 17108 ?        S    Apr03   0:01 /usr/local/apache/bin/httpd -k start
nobody   24481  0.0  0.0  14760  6396 ?        S    Apr03   0:00 /usr/local/apache/bin/httpd -k start
nobody   26089  0.0  0.0  24144 16876 ?        S    Apr03   0:01 /usr/local/apache/bin/httpd -k start
nobody   27842  0.0  0.0  24896 17636 ?        S    Apr03   0:00 /usr/local/apache/bin/httpd -k start
nobody   27843  0.0  0.0  24192 16936 ?        S    Apr03   0:00 /usr/local/apache/bin/httpd -k start
nobody   27911  0.0  0.0  23888 16648 ?        S    Apr03   0:01 /usr/local/apache/bin/httpd -k start
```

```
nobody   28280  0.0  0.0  24664 17256 ?        S    Apr03   0:00 /usr/local/apache/bin/httpd -k start
nobody   30404  0.0  0.0  24360 17112 ?        S    Apr03   0:00 /usr/local/apache/bin/httpd -k start
nobody   31895  0.0  0.0  14760  6296 ?        S    Apr03   0:00 /usr/local/apache/bin/httpd -k start
root     31939  0.0  0.0   1848   548 pts/0    R+   Apr03   0:00 grep http
```

(I deleted about half of the "`httpd -k start`" lines from that output manually to save a little space.)

Similarly, here's how you can find all the Java processes running on your system using the `ps` and `grep` commands in a Unix pipeline:

```
ps auxwww | grep -i java
```

In this example I've piped the output of the `ps auxwww` command into my grep command. The grep command only prints the lines that have the string "java" in them; all other lines from the `ps` command are not printed.

One way to find all the sub-directories in the current directory is to mix the Linux `ls` and `grep` commands together in a pipe, like this:

```
ls -al | grep '^d'
```

Here I'm using grep to list only those lines where the first character in the line is the letter `d`.

Using the Linux grep command to search for multiple patterns at one time (egrep)

You can use a different version of the grep command to search for multiple patterns at one time. To do this, just use the `egrep` command instead of grep, like this:

```
egrep 'score|nation|liberty|equal' gettysburg-address.txt
```

This Unix egrep command searches the file named *gettysburg-address.txt* for the four strings shown (score, nation, liberty, and equal). It returns any lines from the file that contain any of those words.

I should also note that "egrep" stands for "extended grep", and as you can see, it lets you do things like searching for multiple patterns at one time.

Searching for regular expressions (regex patterns) with grep

Of course the Linux grep command is much more powerful than this, and can handle very powerful regular expressions (regex patterns). In a simple example, suppose you want to search for the strings "Foo" or "Goo" in all files in the current directory. That grep command would be:

```
grep '[FG]oo' *
```

If you want to search for a sequence of three integers with grep you might use a command like this:

```
grep '[0-9][0-9][0-9]' *
```

This next grep command searches for all occurrences of the text string `fred` within the */etc/passwd* file, but also requires that the "f" in the name "fred" be in the first column of each record (that's what the caret character tells grep). Using this more-advanced search, a user named "alfred" would not be matched, because the letter "a" will be in the first column:

```
grep '^fred' /etc/passwd
```

Regular expressions can get much, much more complicated (and powerful) than this, so I'll just leave it here for now.

Display only filenames with a grep search

If you're looking through a lot of files for a pattern, and you just want to find the names of the files that contain your pattern (or "patterns", as shown with egrep) -- but don't want to see each individual grep pattern match -- just add the `-l` (lowercase letter L) to your grep command, like this:

```
grep -l StartInterval *.plist
```

This command doesn't show every line in every file that contains the string "StartInterval"; it just shows the names of all the files that contain this string, like this:

```
com.apple.atrun.plist
com.apple.backupd-auto.plist
com.apple.dashboard.advisory.fetch.plist
com.apple.locationd.plist
org.amavis.amavisd_cleanup.plist
```

Of course you can also combine grep command arguments, so if you didn't happen to know how to capitalize "StartInterval" in that previous example, you could just add the `-i` argument to ignore case, like this:

```
grep -il startinterval *.plist
```

and that would have worked just fine as well, returning the same results as the previous grep command example.

Showing matching line numbers with Linux grep

To show the line numbers of the files that match your grep command, just add the `-n` option, like this:

```
grep -n we gettysburg-address.txt
```

Searching my sample *gettysburg-address.txt* file, I get the following output from this command:

```
9:Now we are engaged in a great civil war,
22:that we should do this.
24:But in a larger sense we can not dedicate -
25:we can not consecrate -
26:we can not hallow this ground.
29:have consecrated it far above our poor power
33:what we say here,
43:we take increased devotion to that cause
46:that we here highly resolve that these dead
```

grep before/after - Showing lines before or after your grep pattern match

After a recent comment, I just learned that you can display lines before or after your grep pattern match, which is also very cool. To display five lines before the phrase "the living" in my sample document, use the -B argument, like this:

```
grep -B 5 "the living" gettysburg-address.txt
```

This grep command example returns this output:

```
The world will little note,
nor long remember,
what we say here,
```

```
but can never forget what they did here.

It is for us, the living,
```

Similarly, to show the five lines after that same search phrase, use the -A argument with your Unix grep command, like this:

```
grep -A 5 "the living" gettysburg-address.txt
```

This grep "after" command returns the following output:

```
It is for us, the living,
rather to be dedicated here
to the unfinished work which they have,
thus far, so nobly carried on.
It is rather for us to be here
dedicated to the great task remaining before us -
```

Of course you can use any number after the -A and -B options, I'm just using the number five here as an example.

Power file searching with find and grep

A lot of times I know that the string "foo" exists in a file somewhere in my directory tree, but I can't remember where. In those cases I roll out a power command, a Linux find command that uses grep to search what it finds:

```
find . -type f -exec grep -il 'foo' {} \;
```

This is a special way of mixing the Linux find and grep commands together to search every file in every subdirectory of my current location. It searches for the string "foo" in every file below the current directory, in a case-insensitive manner. This find/grep command can be broken down like this:

- "." means "look in the current directory"
- `-type f` means "look in files only"
- `-exec grep -il foo` means "search for the string 'foo' in a case-insensitive manner, and return the matching line and filename when a match is found
- `{} \;` is a little bizarre syntax that you need to add to the end of your find command whenever you add the `-exec` option. I try to think of it as a placeholder for the filenames the find command finds.

Note that on Mac OS X systems you may be able to use the `mdfind` command instead of this `find/grep` combination command. The `mdfind` command is a command-line equivalent of the Spotlight search functionality.

Related Unix/Linux grep commands and tutorials

We hope you enjoyed this Linux grep command tutorial and our grep examples.

There are at least two other commands related to grep that you should at least be aware of. The `fgrep` command stands for "fast grep", or "fixed strings", depending on who you talk to. The `egrep` command stands for "extended grep", and lets you use even more powerful regular expressions.

The `strings` command is good at finding printable strings in a binary file.

The `locate` command is more related to the `find` command, but I thought I would note that it is good at finding files in the entire filesystem when you know the filename, or part of the filename.

And as I mentioned in the previous section Mac OS X systems have the `mdfind` command. As a practical matter I use plain old `grep` 99% of the time.