How do I find all files containing specific text on Linux?

4625

2388

I'm trying to find a way to scan my entire Linux system for all files containing a specific string of text. Just to clarify, I'm looking for text within the file, not in the file name.

When I was looking up how to do this, I came across this solution twice:

```
find / -type f -exec grep -H 'text-to-find-here' {} \;
```

However, it doesn't work. It seems to display every single file in the system.

Is this close to the proper way to do it? If not, how should I? This ability to find text strings in files would be extraordinarily useful for some programming projects I'm doing.

linux text grep directory find

edited May 21 '17 at 11:40

Peter Mortensen
14.3k 19 88 116

asked Jun 6 '13 at 8:06

Nathan
25.6k 8 28 44

We're looking for long answers that provide some explanation and context. Don't just give a one-line answer; explain why your answer is right, ideally with citations. Answers that don't include explanations may be removed.

19 remember that grep will interpret any `.` as a single-character wildcard, among others. My advice is to alway use either fgrep or egrep. – Walter Tross Oct 28 '13 at 11:54

10 anyway, you were almost there! Just replace `-H` with `-l` (and maybe `grep` with `fgrep`). To exclude files with certain patterns of names you would use `find` in a more advanced way. It's worthwile to learn to use `find`, though. Just `man find`. – Walter Tross Oct 28 '13 at 12:01

5 `find … -exec <cmd> +` is easier to type and faster than `find … -exec <cmd> \;`. It works only if `<cmd>` accepts any number of file name arguments. The saving in execution time is especially big if `<cmd>` is slow to start like Python or Ruby scripts. – hagello Jan 28 '16 at 5:16

To search non-recursively in a given path the command is `grep --include=*.txt -snw "pattern" thepath/*. – Stéphane Laurent Aug 15 '16 at 12:34

@StéphaneLaurent I think you are complicating it too much. Just say `grep "pattern" path/*.txt` – fedorqui Dec 2 '16 at 13:13

42 Answers

1 2 next

8372

Do the following:

```
grep -rnw '/path/to/somewhere/' -e 'pattern'
```

- `-r` or `-R` is recursive,
- `-n` is line number, and
- `-w` stands for match the whole word.
- `-l` (lower-case L) can be added to just give the file name of matching files.

Along with these, `--exclude`, `--include`, `--exclude-dir` flags could be used for efficient searching:

- This will only search through those files which have .c or .h extensions:

```
grep --include=\*.{c,h} -rnw '/path/to/somewhere/' -e "pattern"
```

- This will exclude searching all the files ending with .o extension:

```
grep --exclude=*.o -rnw '/path/to/somewhere/' -e "pattern"
```

- For directories it's possible to exclude a particular directory(ies) through `--exclude-dir` parameter. For example, this will exclude the dirs dir1/, dir2/ and all of them matching *.dst/:

```
grep --exclude-dir={dir1,dir2,*.dst} -rnw '/path/to/somewhere/' -e "pattern"
```

This works very well for me, to achieve almost the same purpose like yours.

For more options check `man grep`.

edited Jan 4 '18 at 3:57

answered Jun 6 '13 at 8:21

rakib_
86.9k 3 13 24

69 use --exclude. like "grep -rnw --exclude=*.o 'directory' -e "pattern" – rakib_ Jun 6 '13 at 8:29

19 I find grep's --include parameter very useful. For example: grep -rnw --include=*.java . -e "whatever I'm looking for" – Lucas A. Nov 14 '13 at 15:43

73 it's worth noting: it seems the `r` option is lazy (traverses depth-first, than stops after the first directory), while `R` is greedy (will traverse the entire tree correctly). – Eliran Malka Mar 24 '15 at 15:09

29 Note(especially for newbies): The quotation marks in the above command are important. – madD7 Dec 22 '15 at 12:37

57 @Eliran Malka `R` en `r` will both traverse directories correctly, but `R` will follow symbolic links. – bzeaman Jul 5 '16 at 8:36

1331

You can use `grep -ilR` :

```
grep -Ril "text-to-find-here" /
```

- `i` stands for ignore case (optional in your case).
- `R` stands for recursive.
- `l` stands for "show the file name, not the result itself".
- `/` stands for starting at the root of your machine.

edited Feb 23 '16 at 10:02

answered Jun 6 '13 at 8:08

fedorqui
180k 56 369 412

78 Based on my experience, the `-i` makes it slow down a lot, so don't use it if not necessary. Test it in a certain dir and then generalise. It should be completed within few minutes. I think a regular expression would make it slower. But my comments are based on suppositions, I suggest you to test it with `time` in front of the line. – fedorqui Jun 6 '13 at 8:14

4 Yes, `/*` stands for that. Anyway I just tested it and noticed that just `/` works. – fedorqui Jun 6 '13 at 8:15

10 If you are not searching using a regex you can use fgrep in place of grep on most systems. – markle976 Sep 28 '13 at 14:49

8 Yes @markle976, in fact from man grep: `fgrep is the same as grep -F -> Interpret PATTERN as a list of fixed strings .` – fedorqui Sep 30 '13 at 8:23

16 You can replace / with path to directory `grep -Ril "text-to-find-here" ~/sites/` or use . for current di... Black Jan 28 '16 at 12:19

fedorqui
180k 56 369 412

Current picture: Love democracy. We deserve a better country, we deserve dialogue, negotiation, debate. We do not want political prisoners, we do not want violence. Free political prisoners from Catalonia.
Contact me at: <my name>.se@gmail.com. I love getting emai...

294

You can use ack. It is like grep for source code. You can scan your entire file system with it.

Just do:

```
ack 'text-to-find-here'
```

In your root directory.

You can also use regular expressions, specify the filetype, etc.

UPDATE

I just discovered The Silver Searcher, which is like ack but 3-5x faster than it and even ignores patterns from a `.gitignore` file.

edited Mar 12 '15 at 12:31

RAJ
8,701 1 26 56

answered Jun 6 '13 at 8:26

EarlOfEgo
8,691 5 36 51

55 Very useful, simple and fast. Warning: "On Debian-derived distros, ack is packaged as "ack-grep" because "ack" already existed" (from beyondgrep.com/install). You may end up running a Kanji code converter on those Linuxes... – Jose_GD Sep 20 '13 at 13:32

11 ack or ack-grep has nice highlights, but find+grep when proper used is much better in performance – Sławomir Lenart Feb 11 '15 at 9:00

11 Note that ripgrep is faster than anything else mentioned here, including The Silver Searcher and plain 'ol grep. See this blog post for proof. – Radon Rosborough Oct 14 '17 at 4:01

169

You can use:

```
grep -r "string to be searched"  /path/to/dir
```

The `r` stands for recursive and so will search in the path specified and also its sub-directories. This will tell you the file name as well as print out the line in the file where the string appears.

Or a command similar to the one you are trying (example:) for searching in all javascript files (*.js):

```
find . -name '*.js' -exec grep -i 'string to search for' {} \; -print
```

This will print the lines in the files where the text appears, but it does not print the file name.

In addition to this command, we can write this too: **grep -rn "String to search" /path/to/directory/or/file** *-r: recursive search n: line number will be shown for matches*

edited Apr 2 '18 at 8:35

Vivek Ranjan
45 6

answered Mar 14 '14 at 23:29

learner_19
2,501 1 14 8

1 Thanx for the find version. My grep version (busybox for NAS) hasn't the -r option, i really needed another solution! – j.c Sep 2 '16 at 10:34

3 Thank you for the 'find' version! It is so important to be able to filter by *'.js' or '*.txt', etc. Nobody wants to spend hours waiting for grep to finish searching all the multi-gigabyte videos from the last family vacation, even if the command is easier to type. – mightypile Aug 16 '17 at 15:10

You can use this:

102

```
grep -inr "Text" folder/to/be/searched/
```

edited Jul 31 '13 at 14:09

Sudipta
3,268 1 18 38

answered Jul 31 '13 at 13:44

A R
1,529 2 12 31

10 easiest, verbose, recursive and case insensitive. thumbs up. – Francesco Casula Apr 9 '15 at 12:44

if you add -A3 is even better – albanx Feb 24 '16 at 10:43

List of file names containing a given text

62

First of all, I believe you have used `-H` instead of `-l`. Also you can try adding the text inside quotes followed by `{} \`.

```
find / -type f -exec grep -l "text-to-find-here" {} \;
```

Example

Let's say you are searching for files containing specific text "Apache License" inside your directory. It will display results somewhat similar to below (output will be different based on your directory content).

```
bash-4.1$ find . -type f -exec grep -l "Apache License" {} \;
./net/java/jvnet-parent/5/jvnet-parent-5.pom
./commons-cli/commons-cli/1.3.1/commons-cli-1.3.1.pom
./io/swagger/swagger-project/1.5.10/swagger-project-1.5.10.pom
./io/netty/netty-transport/4.1.7.Final/netty-transport-4.1.7.Final.pom
./commons-codec/commons-codec/1.9/commons-codec-1.9.pom
./commons-io/commons-io/2.4/commons-io-2.4.pom
bash-4.1$
```

Remove case sensitiveness

Even if you are not use about the case like "text" vs "TEXT", you can use the `-i` switch to ignore case. You can read further details here.

Hope this helps you.

edited Oct 7 '17 at 5:54

answered Nov 9 '13 at 13:18

lkamal
2,177 12 23

2 Which is what this command does: `find` will pass all the paths it finds to the command `grep -l "text-to-find-here" <file found>"` . You may add restrictions to the file name, e.g. `find / -iname "*.txt"` to search only in files which name ends in `.txt` – Mene Apr 20 '17 at 13:46

1 @Auxiliary - included a sample output to avoid any confusion for the readers. – lkamal Oct 7 '17 at 5:56

2 @Mene It's a truly sad state that Auxiliary's comment has more votes than yours...even if their comment is from 2014 and yours is 2017 that their comment has 6 when it should have exactly 0 and yours only had one (now two) isn't something I'd like to believe. – Pryftan May 1 '18 at 23:01

@Mene That being said `-iname` is case-insensitive which means it would also find .TXT files, for example, as well as TxT and TXt and so on. – Pryftan May 1 '18 at 23:04

`grep` (GNU or BSD)

58

You can use `grep` tool to search recursively the current folder, like:

```
grep -r "class foo" .
```

Note: `-r` - Recursively search subdirectories.

You can also use globbing syntax to search within specific files such as:

```
grep "class foo" **/*.c
```

Note: By using globbing option (`**`), it scans all the files recursively with specific extension or pattern. **To enable this syntax, run:** `shopt -s globstar` **.** You may also use `**/*.*` for all files (excluding hidden and without extension) or any other pattern.

If you've the error that your argument is too long, consider narrowing down your search, or use `find` syntax instead such as:

```
find . -name "*.php" -execdir grep -nH --color=auto foo {} ';'
```

Alternatively use `ripgrep` .

`ripgrep`

If you're working on larger projects or big files, you should use `ripgrep` instead, like:

```
rg "class foo" .
```

Checkout the docs, installation steps or source code on the GitHub project page.

It's much quicker than any other tool like GNU/BSD `grep` , `ucg` , `ag` , `sift` , `ack` , `pt` or similar, since it is built on top of Rust's regex engine which uses finite automata, SIMD and aggressive literal optimizations to make searching very fast.

It supports ignore patterns specified in `.gitignore` files, so a single file path can be matched against multiple glob patterns simultaneously.

You can use the common parameters such as:

- `-i` - Insensitive searching.
- `-I` - Ignore the binary files.
- `-w` - Search for the whole words (in opposite of partial word matching).
- `-n` - Show the line of your match.
- `-C` / `--context` (e.g. `-C5`) - Increases context, so you see the surrounding code .
- `--color=auto` - Mark up the matching text.
- `-H` - Displays filename where the text is found.
- `-c` - Displays count of matching lines. Can be combined with `-H` .

edited Apr 10 '18 at 13:55

answered May 9 '15 at 10:11

kenorb

77.9k 33 444 456

1 I also find extended globbing useful. But keep in mind that if there are really huge number of files, you can get a "Argument list too long" error. (Simple globbing is also prone to this kind of error). – Yoory N. Nov 30 '17 at 6:47

For inhaling a whole file system, rg is gonna be far less painful than almost any other tool. – I.k Apr 23 at 6:11

53

If your `grep` doesn't support recursive search, you can combine `find` with `xargs` :

```
find / -type f | xargs grep 'text-to-find-here'
```

I find this easier to remember than the format for `find -exec` .

This will output the filename and the content of the matched line, e.g.

```
/home/rob/file:text-to-find-here
```

Optional flags you may want to add to `grep` :

- `-i` - case insensitive search
- `-l` - only output the filename where the match was found
- `-h` - only output the line which matched (not the filename)

edited Aug 12 '15 at 9:19

answered Jun 20 '14 at 8:49

RobEarl

7,146 6 28 48

3 This is equivalent to `grep 'text-to-find-here'` without file name if `find` does not find anything. This will hang and wait for user input! Add `--no-run-if-empty` as an option to `xargs` . – hagello Jan 28 '16 at 5:46

3 This combination of find and xargs does not work as intended if file or directory names contain spaces (characters that xargs interprets as separators). Use `find … -exec grep … +` . If you insist on using find together with xargs, use `-print0` and `-0` . – hagello Jan 28 '16 at 5:50

40

```
grep -insr "pattern" *
```

- `i` : Ignore case distinctions in both the PATTERN and the input files.
- `n` : Prefix each line of output with the 1-based line number within its input file.
- `s` : Suppress error messages about nonexistent or unreadable files.
- `r` : Read all files under each directory, recursively.

edited Mar 8 '16 at 7:39 — Fabio Poloni — 6,044 3 34 68

answered Feb 26 '16 at 5:47 — enfinet — 540 5 13

3 Can you explain how your answer improves upon the other answers, or how it is sufficiently different from them? – Amos M. Carpenter Feb 26 '16 at 6:10

not much complex to remember, will cover all patterns(case-senstivity -> off, includes file-names and line number and will do recursively search etc) and using "*" in the end will search all directories (no need to specify any path or directory name). – enfinet Feb 26 '16 at 6:15

Sorry, I should've been clearer: it would be great if you could include that explanation in your answer. As it stands, especially with so many other similar answers already, it is hard to see from such a short answer what the benefit of trying *it* over the accepted answer or one of the upvoted ones would be. – Amos M. Carpenter Feb 26 '16 at 6:35

6 @AmosM.Carpenter One thing I love about this answer is pointing out the suppress argument, which can help filter out noise that doesn't matter to getting the results we actually want. Grep prints errors like, "Function not implemented", "Invalid Argument", "Resource unavailable", etc. etc on certain "files". – leetNightshade Feb 20 '17 at 5:58

@leetNightshade: I'm assuming you're addressing your comment to me because I asked for an explanation on the sparse original post. Please see Fabio's great revision for my previous comments to make sense. – Amos M. Carpenter Feb 20 '17 at 11:59

32

> How do I find all files containing specific text on Linux? (...)
>
> I came across this solution twice:
>
> ```
> find / -type f -exec grep -H 'text-to-find-here' {} \;
> ```

If using **find** like in your example, better add `-s` (`--no-messages`) to `grep` , and `2>/dev/null` at the end of the command to avoid lots of *Permission denied* messages issued by `grep` and `find` :

```
find / -type f -exec grep -sH 'text-to-find-here' {} \; 2>/dev/null
```

find is the standard tool for searching files - combined with grep when looking for specific text - on Unix-like platforms. The find command is often combined with xargs, by the way.

Faster and easier tools exist for the same purpose - see below. Better try them, *provided they're available on your platform*, of course:

Faster and easier alternatives

RipGrep - fastest search tool around:

```
rg 'text-to-find-here' / -l
```

The Silver Searcher:

```
ag 'text-to-find-here' / -l
```

ack:

```
ack 'text-to-find-here' / -l
```

Note: You can add `2>/dev/null` to these commands as well, to hide many error messages.

Warning: *unless you really can't avoid it, don't search from '/' (the root directory) to avoid a long and inefficient search! So in the examples above, you'd better replace '/' by a sub-directory name, e.g. "/home" depending where you actually want to search...*

edited Jun 2 '18 at 17:21

answered Nov 25 '15 at 14:16 — Bludzee — 1,796 5 25 35

'find is the standard tool for searching files containing specific text on Unix-like platforms' seems rather ambiguous to me. Even besides recursive `grep` `find` doesn't directly search the inside of files for text. And maybe those additional tools are useful to some but old timers and those whoa are well accustomed to e.g. `grep` wouldn't give them any time at all (well I certainly won't). Not saying they're useless though. – Pryftan May 1 '18 at 23:36

"....containing specific text..." : this part of the sentence was not accurate (because it's not find itself that deals with this part of the search). Edited. Thanks. – Bludzee Jun 1 '18 at 9:21

Glad to be of help! The only thing else at a very very quick glance is changing the word *folder* to *directory* but I know that's a crusade of mine I will never win completely. Not giving up though... – Pryftan Jun 1 '18 at 16:13

Why not "directory" instead of "folder", but why ? Please share your "crusade" ! – Bludzee Jun 1 '18 at 17:00

I'm saying use directory instead! Referring to: **you'd better replace '/' by a sub-folder name** And it's a pet peeve of mine.. esp since even Windows used to call it 'directory'. Ah..maybe you got that. Why? Well because that's what it's called. It's also called that at the file system level. And look at it this way: was it ever called (for DOS) `fol` ? No of course not; it was called `dir` (and I believe it still is). Folder is a thing contrived for (I guess) user friendliness though in this case it's maybe dumbing it down for less 'advanced' users? – Pryftan Jun 2 '18 at 0:15

29

Try:

```
find . -name "*.txt" | xargs grep -i "text_pattern"
```

edited Apr 17 '15 at 15:33
kenorb
77.9k 33 444 456

answered Dec 10 '14 at 5:47
venkat
299 3 2

5 This is actually a prime example of when NOT to use `xargs` like that .. consider this. `echo "file bar.txt has bar" > bar.txt; echo "file foo bar.txt has foo bar" > "foo bar.txt"; echo "You should never see this foo" > foo; find . -name "*.txt" | xargs grep -i foo # ./foo:You should never see this foo` . The `xargs` here matched the WRONG file and did NOT match the intended file. Either use a `find .. -print0 | xargs -0 ...` but that's a useless use of a pipe or better `find ... -exec grep ... {} +` – shalomb Oct 11 '16 at 20:10

29

There's a new utility called **The Silversearcher**

```
sudo apt install silversearcher-ag
```

It works closely with Git and other VCS. So you won't get anything in a *.git* or another directory.

You can simply use

```
ag "Search query"
```

And it will do the task for you!

edited yesterday

answered Nov 3 '16 at 21:30
Neil Agarwal
432 4 8

28

Use `pwd` to search from any directory you are in, recursing downward

```
grep -rnw `pwd` -e "pattern"
```

Update Depending on the version of grep you are using, you can omit `pwd` . On newer versions `.` seems to be the default case for grep if no directory is given thus:

```
grep -rnw -e "pattern"
```

or

```
grep -rnw "pattern"
```

will do the same thing as above!

edited Feb 2 '17 at 12:29

answered May 28 '16 at 12:47
mahatmanich
6,926 2 45 67

3 using `pwd` is not necessary at all, since it is the default. `grep -rnw "pattern"` suffices. – fedorqui Dec 2 '16 at 13:17

and in fact the `grep -rnw` and similar is what was answered like three years ago, I don't see how this answer is adding value. – fedorqui Dec 2 '16 at 14:03

The selected answer does not show the default pattern, and 5 peoples seemed to have found it useful – mahatmanich Dec 14 '16 at 8:27

What do you mean with "default pattern"? The accepted answer contains `grep -rnw '/path/to/somewhere/' -e "pattern"` which is what you have here. 5 votes after 2.3M visits does not mean that much. – fedorqui Dec 14 '16 at 8:45

I agree :-) what I was missing in the original answer is the use case that you don't have to give a path at all or to search the current directory recursively which is not reflected in the accepted answer. Thus it was a good learning experience about grep to dig a bit deeper. – mahatmanich Dec 14 '16 at 14:05

19

`grep` can be used even if we're not looking for a string.

Simply running,

```
grep -RIl "" .
```

will print out the path to all text files, i.e. those containing only printable characters.

edited May 25 '17 at 0:01
Peter Mortensen
14.3k 19 88 116

answered Apr 9 '14 at 19:51
Alex Jasmin
34.2k 5 65 61

2 I don't see how this is better than using a mere `ls` or `find` (for the recursive) – fedorqui Dec 2 '16 at 13:15

17

Here are the several list of commands that can be used to search file.

```
grep "text string to search" directory-path

grep [option] "text string to search" directory-path

grep -r "text string to search" directory-path

grep -r -H "text string to search" directory-path

egrep -R "word-1|word-2" directory-path

egrep -w -R "word-1|word-2" directory-path
```

answered Feb 1 '14 at 5:47
Atul Arvind
10.8k 5 37 52

5 what is this adding to the existing answers? – fedorqui Dec 2 '16 at 13:14

@fedorqui `egrep` is equivalent to `grep -E` and it means `--extended-regexp` you can find details here unix.stackexchange.com/a/17951/196072 – omerhakanbilici Jul 25 '18 at 11:00

15

```
find /path -type f -exec grep -l "string" {} \;
```

Explanation from comments

find is a command that lets you find files and other objects like directories and links in subdirectories of a given path. If you don't specify a mask that filesnames should meet, it enumerates all directory objects.

```
-type f specifies that it should proceed only files, not directories etc.
-exec grep specifies that for every found file, it should run grep command, passing its
filename as an argument to it, by replacing {} with the filename
```

edited Nov 26 '14 at 13:12
JuanZe
7,024 37 55

answered Jul 2 '14 at 7:18
Vinod Joshi
6,183 41 46

15

Try:

```
find / -type f -exec grep -H 'text-to-find-here' {} \;
```

which will search all file systems, because `/` is the root folder.

For home folder use:

```
find ~/ -type f -exec grep -H 'text-to-find-here' {} \;
```

For current folder use:

```
find ./ -type f -exec grep -H 'text-to-find-here' {} \;
```

edited May 9 '15 at 9:49
kenorb

answered May 4 '15 at 19:11
user4863663

77.9k 33 444 456 169 1 2

Perhaps the details on differences of folders are obvious to many ...but also very helpful for newbies. +1 – nilon Oct 17 '16 at 18:07

1 what is this adding to the existing answers? – fedorqui Dec 2 '16 at 13:16

Call it my crusade but the word is 'directory'. This isn't Windows (which used to use 'directory' anyway - pre 9x). Please stop saying 'folder'. As for your last command you don't even need the '/' just FYI. – Pryftan May 1 '18 at 23:12

15

Hope this is of assistance...

Expanding the `grep` a bit to give more information in the output, for example, to get the line number in the file where the text is can be done as follows:

```
find . -type f -name "*.*" -print0 | xargs --null grep --with-filename --line-number --
no-messages --color --ignore-case "searthtext"
```

And if you have an idea what the file type is you can narrow your search down by specifying file type extensions to search for, in this case `.pas` OR `.dfm` files:

```
find . -type f \( -name "*.pas" -o -name "*.dfm" \) -print0 | xargs --null grep --with-
filename --line-number --no-messages --color --ignore-case "searchtext"
```

Short explanation of the options:

1. `.` in the `find` specifies from the current directory.
2. `-name " *.* "` : for all files (-name " `*.pas` " -o -name " `*.dfm` ") : Only the `*.pas` OR `*.dfm` files, OR specified with `-o`
3. `-type f` specifies that you are looking for files
4. `-print0` and `--null` on the other side of the `|` (pipe) are the crucial ones, passing the filename from the `find` to the `grep` embedded in the `xargs` , allowing for the passing of filenames WITH spaces in the filenames, allowing grep to treat the path and filename as one string, and not break it up on each space.

edited May 25 '17 at 0:03 — Peter Mortensen 14.3k 19 88 116

answered Jan 28 '15 at 6:42 — Gert van Biljon 171 1 6

`-name '*.*'` isn't what you say; it wouldn't pick up on a file called 'file' because the pattern doesn't equate to that (no .ext); `*` would however (well . files aside). But there's another thing: if you want all files why bother specifying a file name in the first place? No other comment - except that it's nice to know that there still are people who don't use the MS terminology 'folder' (which really after saying it enough I wouldn't add but I wanted to point out the slightly incorrect statement you made with file names - as well as the redundancy/uselessness in the case of 'all'). – Pryftan May 1 '18 at 23:25

15

Silver Searcher is a terrific tool, but ripgrep may be even better.

It works on Linux, Mac and Windows, and was written up on Hacker News a couple of months ago (this has a link to Andrew Gallant's Blog which has a GitHub link):

Ripgrep – A new command line search tool

edited May 25 '17 at 0:10 — Peter Mortensen 14.3k 19 88 116

answered Dec 13 '16 at 5:48 — AAAfarmclub 1,278 11 11

15

A Simple `find` can work handy. alias it in your `~/.bashrc` file:

```
alias ffind find / -type f | xargs grep
```

Start a new terminal and issue:

```
ffind 'text-to-find-here'
```

edited May 25 '17 at 0:15 — Peter Mortensen 14.3k 19 88 116

answered Mar 22 '17 at 12:30 — danglingpointer 2,943 3 14 30

14

I wrote a Python script which does something similar. This is how one should use this script.

```
./sniff.py path pattern_to_search [file_pattern]
```

The first argument, `path`, is the directory in which we will search recursively. The second argument, `pattern_to_search`, is a regular expression which we want to search in a file. We use the regular expression format defined in the Python `re` library. In this script, the `.` also matches newline.

The third argument, `file_pattern`, is optional. This is another regular expression which works on a filename. Only those files which matches this regular expression will be considered.

For example, if I want to search Python files with the extension `py` containing `Pool(` followed by word `Adaptor`, I do the following,

```
./sniff.py . "Pool(.*?Adaptor"  .*py
./Demos/snippets/cubeMeshSigNeur.py:146
./Demos/snippets/testSigNeur.py:259
./python/moose/multiscale/core/mumbl.py:206
./Demos/snippets/multiComptSigNeur.py:268
```

And voila, it generates the path of matched files and line number at which the match was found. If more than one match was found, then each line number will be appended to the filename.

edited Jun 30 '14 at 17:59
Peter Mortensen
14.3k 19 88 116

answered Jan 6 '14 at 12:56
Dilawar
2,501 7 33 49

13

Use:

```
grep -c Your_Pattern *
```

This will report how many copies of your pattern are there in each of the files in the current directory.

edited May 25 '17 at 0:13
Peter Mortensen
14.3k 19 88 116

answered Jan 7 '17 at 7:59
Dr_Hope
389 3 15

12

To search for the string and output just that line with the search string:

```
for i in $(find /path/of/target/directory -type f); do grep -i "the string to look for"
"$i"; done
```

e.g.:

```
for i in $(find /usr/share/applications -type f); \
do grep -i "web browser" "$i"; done
```

To display filename containing the search string:

```
for i in $(find /path/of/target/directory -type f); do if grep -i "the string to look
for" "$i" > /dev/null; then echo "$i"; fi; done;
```

e.g.:

```
for i in $(find /usr/share/applications -type f); \
do if grep -i "web browser" "$i" > /dev/null; then echo "$i"; \
fi; done;
```

answered Jan 25 '14 at 11:08
user3124504

1 I see only downside compared to using `find … -exec grep 'str' {} \;` (if you have to use `find` at all). – phk Oct 7 '16 at 16:14

1 This would break horribly if any of the files found by `find` contained spaces .. you could end up `grepping` the wrong files and/or missing the right files altogether. Just use `find ... -exec grep ...` if you have a need to use `find` .. but in this case a `grep -r ...` suffices. – shalomb Oct 11 '16 at 20:19

1 what is the point of using a loop over the results of find to then grep? This gets unnecessarily complicated. – fedorqui Dec 2 '16 at 13:17

12

There is an `ack` tool that would do exactly what you are looking for.

http://linux.die.net/man/1/ack

```
ack -i search_string folder_path/*
```

You may ignore `-i` for case sensitive search

edited Jul 13 '17 at 0:17
Daniel
3,303 2 27 39

answered Aug 17 '16 at 3:31
Pal
502 5 17

2 What is this adding to the existing answers? This was suggested more than three years ago already. – fedorqui Dec 2 '16 at 13:20

1 @fedorqui 1)no piping! 2)Use regular expressions 3)Get line numbers, file name with relative path, highlighted text etc. useful for editing after the search e.g "vim +lineno path/file.cpp" will get you right at the line no of interest. See the output of the command "ack include\|hpp" that searches "include" or "hpp" keywords under my search folder and subfolders. I hope the point is clear. Here is the sample output(Can't show the keyword highlights with simple text) process/child.hpp 11:boost/process/child.hpp process/all.hpp 21:#include <boost/process/execute.hpp> – Pal Jul 11 '17 at 15:57

12

grep is your good friend to achieve this.

```
grep -r <text_fo_find> <directory>
```

if you don't care about the case of the text to find then use

```
grep -ir <text_to_find> <directory>
```

answered Oct 24 '17 at 3:07
Prash
318 3 6

In my case it looks like it searches everywhere even if I do specify the directory – Pathros Mar 20 '18 at 16:30

@Pathros Probably to do with recursion enabled and what directory you specify. Put another way recursion does change things in that way. – Pryftan May 1 '18 at 23:38

@Pathros Oh and if there are any `-` s in the search string you'll want to pass in `--` to grep first; that can cause interesting side effects otherwise! – Pryftan Jun 2 '18 at 0:25

11

All previous answers suggest grep and find. But there is another way: Use Midnight Commander

It is a free utility (30 years old, proven by time) which is visual without being GUI. It has tons of functions, and finding files is just one of them.

edited Apr 24 at 15:28
Peter Mortensen
14.3k 19 88 116

answered Jun 22 '17 at 16:27
Peter M.
745 9 12

ranger would be in the same idea – nilon Jul 12 '17 at 20:03

10

The below command will work fine for this approach:

```
find ./ -name "file_pattern_name"  -exec grep -r "pattern" {} \;
```

edited Feb 16 '16 at 23:47
Peter Mortensen
14.3k 19 88 116

answered Dec 12 '15 at 6:14
Pradeep Goswami
1,008 10 21

2 what is the point of using `find` and then `grep -r` ? They are meant for the same, so this is redundant. – fedorqui Dec 23 '15 at 17:02

ohh!! corrected , Actually find is for running grep on filtered files and not all, thanks – Pradeep Goswami Dec 30 '15 at 15:25

2 still, this does not make any sense, you can filter with `find .` – fedorqui Dec 3 '16 at 23:00

10

Avoid the hassle and install ack-grep. It eliminates a lot of permission and quotation issues.

```
apt-get install ack-grep
```

Then go to the directory you want to search and run the command below

```
cd /
ack-grep "find my keyword"
```

answered Mar 23 '16 at 3:41
Kareem
2,798 29 26

Try this:

10

```
find . | xargs grep 'word' -sl
```

edited May 25 '17 at 0:02

Peter Mortensen
14.3k 19 88 116

answered Dec 15 '14 at 10:10

Tayab Hussain
355 2 13

4 this is far slower than the grep solution – amine Dec 22 '14 at 16:58

@amine Yeah rather than using *grep* directly it pipes all the files *find* finds to xargs running *grep* on it. I'm sure you understand that but just to add to those who might not. The command here is .. I can't atm think of a good analogy but it's adding a lot of unnecessary and harmless overhead. – Pryftan Jun 2 '18 at 1:34

I am fascinated by how simple grep makes it with 'rl'

10

```
grep -rl 'pattern_to_find' /path/where/to/find

-r to find recursively file / directory inside directories..
-l to list files matching the 'pattern'
```

> Use '-r' without 'l' to see the file names followed by **text in which the pattern is found**!

```
grep -r 'pattern_to_find' /path/where/to/find
```

Works just perfect..

Hope it helps!

edited Sep 4 '17 at 6:13

answered Aug 8 '17 at 9:38

nitinr708
900 2 15 24

This also works in Git Bash (Windows). – Peter Mortensen Apr 24 at 15:32

But it implies every file must searched (no filter on the file name or file extension level, like `.txt`). Or is there a way to do that? – Peter Mortensen Apr 24 at 16:17

1 2 next