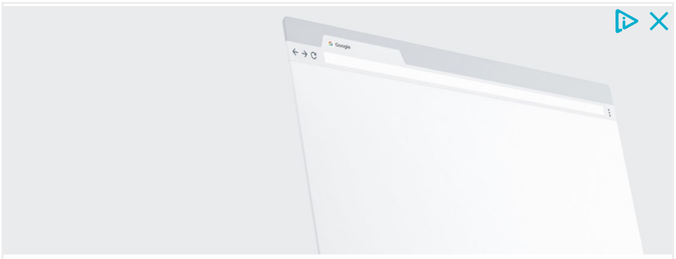
25 simple examples of Linux find command

By Silver Moon | August 12, 2013 43 Comments

Linux find command



The Linux **find command** is a very useful and handy command to search for files from the command line. It can be used to find files based on various search criterias like permissions, user ownership, modification date/time, size etc. In this post we shall learn to use the find command along with various options that it supports.

The find command is available on most linux distros by default so you do not have to install any package. The find command is an essential one to learn, if you want to get super productive with the command line on linux.

The basic syntax of the find command looks like this

```
$ find location comparison-criteria search-t
erm
```

1. List all files in current and sub directories

This command lists out all the files in the current directory as well as the subdirectories in the current directory.

```
$ find
.
./abc.txt
./subdir
./subdir/how.php
./cool.php
```

The command is same as the following

```
$ find .
$ find . -print
```

2. Search specific directory or path

The following command will look for files in the test directory in the current directory. Lists out all files by default.

```
$ find ./test
./test
./test/abc.txt
./test/subdir
```

```
./test/subdir/how.php
./test/cool.php
```

The following command searches for files by their name.

```
$ find ./test -name "abc.txt"
./test/abc.txt
```

We can also use wildcards

```
$ find ./test -name "*.php"
./test/subdir/how.php
./test/cool.php
```

Note that all sub directories are searched recursively. So this is a very powerful way to find all files of a given extension.

Trying to search the "/" directory which is the root, would search the entire file system including mounted devices and network storage devices. So be careful. Of course you can press Ctrl + c anytime to stop the command.

```
When specifying the directory ("./test" in this ex
ample), its fine to omit the trailing slash. Howev
er, if the directory is actually a symlink to some
 other location then you MUST specify the trailing
 slash for it to work properly (find ./test/ ...)
```

Ignore the case

It is often useful to ignore the case when searching for file names. To ignore the case, just use the "iname" option instead of the "name" option.

```
$ find ./test -iname "*.Php"
./test/subdir/how.php
./test/cool.php
```

```
Its always better to wrap the search term (name pa
rameter) in double or single quotes. Not doing so
will seem to work sometimes and give strange resul
ts at other times.
```

3. Limit depth of directory traversal

The find command by default travels down the entire directory tree recursively, which is time and resource consuming. However the depth of directory travesal can be specified. For example we don't want to go more than 2 or 3 levels down in the sub directories. This is done using the maxdepth option.

```
$ find ./test -maxdepth 2 -name "*.php"
./test/subdir/how.php
./test/cool.php

$ find ./test -maxdepth 1 -name *.php
./test/cool.php
```

The second example uses maxdepth of 1, which means it will not go lower than 1 level deep, either only in the current directory.

This is very useful when we want to do a limited search only in the current directory or max 1 level deep sub directories and not the entire directory tree which would take more time.

Just like maxdepth there is an option called mindepth which does what the name suggests, that is, it will go atleast N level deep before searching for the files.

4. Invert match

It is also possible to search for files that do no match a given name or pattern. This is helpful when we know which files to exclude from the search.

```
$ find ./test -not -name "*.php"
./test
./test/abc.txt
./test/subdir
```

So in the above example we found all files that do not have the extension of php, either non-php files. The find command also supports the exclamation mark inplace of not.

```
find ./test ! -name "*.php"
```

5. Combine multiple search criterias

It is possible to use multiple criterias when specifying name and inverting. For example

```
$ find ./test -name 'abc*' ! -name '*.php'
./test/abc.txt
./test/abc
```

The above find command looks for files that begin with abc in their names and do not have a php extension. This is an example of how powerful search expressions can be build with the find command.

OR operator

When using multiple name criterias, the find command would combine them with AND operator, which means that only those files which satisfy all criterias will be matched. However if we need to perform an OR based matching then the find command has the "o" switch.

```
$ find -name '*.php' -o -name '*.txt'
./abc.txt
./subdir/how.php
./abc.php
./cool.php
```

The above command search for files ending in either the php extension or the txt extension.

6. Search only files or only directories

Sometimes we want to find only files or only directories with a given name. Find can do this easily as well.

```
$ find ./test -name abc*
./test/abc.txt
./test/abc

Only files

$ find ./test -type f -name "abc*"
./test/abc.txt

Only directories

$ find ./test -type d -name "abc*"
./test/abc
```

Quite useful and handy!

7. Search multiple directories together

So lets say you want to search inside 2 separate directories. Again, the command is very simple

```
$ find ./test ./dir2 -type f -name "abc*"
./test/abc.txt
./dir2/abcdefg.txt
```

Check, that it listed files from 2 separate directories.

8. Find hidden files

Hidden files on linux begin with a period. So its easy to mention that in the name criteria and list all hidden files.

```
$ find ~ -type f -name ".*"
```

9. Find files with certain permissions

The find command can be used to find files with a specific permission using the "perm" option. The following command searches for files with the permission 0664

```
$ find . -type f -perm 0664
./abc.txt
./subdir/how.php
./abc.php
./cool.php
```

This can be useful to find files with wrong permissions which can lead to security issues. Inversion can also be applied to permission checking.

```
$ find . -type f ! -perm 0777
./abc.txt
./subdir/how.php
./abc.php
./cool.php
```

10. Find files with sgid/suid bits set

The "perm" option of find command accepts the same mode string like chmod. The following command finds all files with permission 644 and sgid bit set.

```
# find / -perm 2644
```

Similarly use 1664 for sticky bit. The perm option also supports using an alternative syntax instead of octal numbers.

```
$ find / -maxdepth 2 -perm /u=s 2>/dev/null
/bin/mount
/bin/su
/bin/ping6
/bin/fusermount
/bin/ping
/bin/umount
/sbin/mount.ecryptfs_private
```

Note that the "2>/dev/null" removes those entries that have an error of "Permission Denied"

11. Find readonly files

Find all Read Only files.

```
$ find /etc -maxdepth 1 -perm /u=r
/etc
/etc/thunderbird
/etc/brltty
/etc/dkms
/etc/phpmyadmin
... output truncated ...
```

12. Find executable files

The following command will find executable files

```
$ find /bin -maxdepth 2 -perm /a=x
/bin
/bin/preseed_command
/bin/mount
/bin/zfgrep
/bin/tempfile
... output truncated ...
```

13. Find files owned to particular user

To find all or single file called tecmint.txt under /root directory of owner root.

```
$ find . -user bob
.
./abc.txt
./abc
./subdir
./subdir/how.php
./abc.php
```

We could also specify the name of the file or any name related criteria along with user criteria

```
$ find . -user bob -name '*.php'
```

Its very easy to see, how we can build up criteria after criteria to narrow down our search for matching files.

14. Search files belonging to group

Find all files that belong to a particular group.

```
# find /var/www -group developer
```

Did you know you could search your home directory by using the ~ symbol ?

```
$ find ~ -name "hidden.php"
```

Easy!!

Search file and directories based on modification date and time

Another great search criteria that the find command supports is modification and accessed date/times. This is very handy when we want to find out which files were modified as a certain time or date range. Lets take a few examples

15. Find files modified N days back

To find all the files which are modified 50 days back.

```
# find / -mtime 50
```

16. Find files accessed in last N days

Find all files that were accessed in the last 50 days.

```
# find / -atime 50
```

17. Find files modified in a range of days

Find all files that were modified between 50 to 100 days ago.

```
# find / -mtime +50 –mtime -100
```

18. Find files changed in last N minutes.

Find files modified within the last 1 hour.

```
$ find /home/bob -cmin -60
```

19. Files modified in last hour

To find all the files which are modified in last 1 hour.

```
# find / -mmin -60
```

20. Find Accessed Files in Last 1 Hour

To find all the files which are accessed in last 1 hour.

```
# find / -amin -60
```

21. Find files of given size

Search files and directories based on size. To find all 50MB files, use.

```
# find / -size 50M
```

22. Find files in a size range

To find all the files which are greater than 50MB and less than 100MB.

```
$ find / -size +50M -size -100M
```

23. Find largest and smallest files

The find command when used in combination with the ls and sort command can be used to list out the largest files. The following command will display the 5 largest file in the current directory and its subdirectory. This may take a while to execute depending on the total number of files the command has to process.

```
$ find . -type f -exec ls -s {} \; | sort -n -r
| head -5
```

Similary when sorted in ascending order, it would show the smallest files first

```
$ find . -type f -exec ls -s {} \; | sort -n | h
ead -5
```

24. Find empty files and directories

The following command uses the "empty" option of the find command, which finds all files that are empty.

```
# find /tmp -type f -empty
```

To file all empty directories use the type "d".

```
$ find ~/ -type d -empty
```

Really very simple and easy

Some advanced operations

The find command not only finds files based on a certain criteria, it can also act upon those files using any linux command. For example, we might want to delete some files.

Here are some quick examples

25. List out the found files

Lets say we found files using find command, and now want to list them out as the ls command would have done. This is very easy.

```
$ find . -exec ls -ld {} \;
drwxrwxr-x 4 enlightened enlightened 4096 Aug 11
 19:01 .
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16
:25 ./abc.txt
drwxrwxr-x 2 enlightened enlightened 4096 Aug 11
 16:48 ./abc
drwxrwxr-x 2 enlightened enlightened 4096 Aug 11
 16:26 ./subdir
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16
:26 ./subdir/how.php
-rw-rw-r-- 1 enlightened enlightened 29 Aug 11 1
9:13 ./abc.php
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16
:25 ./cool.php
```

26. Delete all matching files or directories

The following command will remove all text files in the tmp directory.

```
$ find /tmp -type f -name "*.txt" -exec rm -f {}
 \;
```

The same operating can be carried out with directories, just put type d, instead of type f.

Lets take another example where we want to delete files larger than 100MB

```
$ find /home/bob/dir -type f -name *.log -size +
10M -exec rm -f {} \;
```

Summary

So that was a quick tutorial on the **linux find command**. The find command is one of the most essential commands on the linux terminal, that enables searching of files very easy. Its a must of all system administrators. So learn it up. Have any questions ? Leave a comment below.

Last Updated On : 17th February 2018

Facebook Twitter Google+

Pinterest

CATEGORY: LINUX COMMANDS

TAGS: **LINUX COMMANDS**

About Silver Moon

Php developer, blogger and Linux enthusiast. He can be reached at **binarytides@gmail.com**. Or find him on **Google+**

View all posts by Silver Moon →

43 thoughts on “ 25 simple examples of Linux find command ”

Emre

May 28, 2018 at 10:10 pm

Good article

Ivan

January 10, 2018 at 9:09 pm

Thanks for the article. It’s really useful. I am wondering how can I use the find command and exclude the files that are using by some processes? I can find if files are used by process by:
find path_to_files -type f -name “some_name” -exec fuser {} \;
The output will show if there are files used by any processes but how can I get output that will show only files that are currently not used by any process?

Aung Myat

[December 27, 2017 at 8:08 am]

It would become more perfect if “Delete all files which are older than X time” in the last. This is a good article. Thanks for sharing.

Greg Faucher

[December 6, 2017 at 8:08 am]

yikes!!

[Betty Ann]

[December 2, 2017 at 9:09 pm]

Your article is very informative. it’s really helpful……

Will Stites

[November 17, 2017 at 2:02 am]

You use the word “criterias”. That’s a mistake. Criteria is already plural. The singular is criterion. Thanks for providing the info here.

Erick

[October 27, 2017 at 9:09 pm]

Excelente presentacion. buenos ejemplos y muy practicos.
Saludos.

PeterM
[October 23, 2017 at 2:02 am]

That was really helpful. Concise and clear article, thank you!

Shaiju
[May 30, 2017 at 5:05 pm]

Thank You..:)

Yasser
[May 7, 2017 at 7:07 pm]

Very nice presentation. Thank you so much

Yogesh Mane
[April 9, 2017 at 12:12 pm]

Very useful find command explanation. Thanks

---

horseandbuggy
[March 28, 2017 at 3:03 pm]

In connection with -o need to mention use of escaped parentheses \(, \) to group the criteria

Note: one criterion, two criteria, no criterias.

Also “Its” is a possessive adjective. “it’s” is short for “it is”.

j

[February 15, 2017 at 10:10 am]

thanks a lot for very useful post =))

---

lopez

[January 23, 2017 at 8:08 pm]

Very useful article! Thank you!

Joe

[October 28, 2016 at 4:04 pm]

You are awesome! this is a treasure trove of helpful information!

---

alehandro alberta

[September 30, 2016 at 11:11 pm]

I used to have similar problems too, but after using"long path tool" everything was solved.

---

stepan raisa

[September 28, 2016 at 6:06 pm]

Please use this software and solve your computer, copy, delete, long path files.

---

hehe
September 24, 2016 at 6:06 am

Thanks for these good examples.

---

albina elvira
September 6, 2016 at 2:02 pm

The path you entered, is too long. Enter a shorter path
File Name could not be found. Check the spelling of the filename,
and verify that the file location is correct.

Johnraf
September 3, 2016 at 9:09 pm

I used to have similar problems too, but after using "long path tool" You can use to solve this problem.

---

albina elvira
August 25, 2016 at 9:09 am

The path you entered, is too long. Enter a shorter path
File Name could not be found. Check the spelling of the filename,
and verify that the file location is correct.

---

barryk desteve
August 21, 2016 at 9:09 pm

Do not worry if you want to remove the blocked files or too long path files from your system, here I suggest a smooth way. Use "Long path tool" software and keep yourself cool.

George K.

July 20, 2016 at 10:10 pm

Great intro! Thanks!
I think I have found a typo, in the Section "Find readonly files": instead of "-perm /u=r" (which means: at least user-readable — compare with the following section), it should be "-perm -u=r". That is, according to the man, '/' means "at least" and '-' means "exactly". I admit though that the man daunted me before I saw this page.

---

alexander forster

June 16, 2016 at 8:08 pm

every command i tried fails with "find: paths must precede expression"
why?

---

naga

June 10, 2016 at 7:07 pm

find . -type f -exec ls -s {} \; | sort -n -r | head -5

what is ls -s {} ??

-s what it will do ??

what about braces "{ } " in that ?

Jim Ward

July 15, 2016 at 9:09 pm

find . -type f -exec ls -s {} \; | sort -n -r | head -5

find = find files
. = directory to start at
-type = only match regular files

-exec = when you find a matching file, run this command
{} = the matching file
\; = the end of the command

So when find matches find foo.txt, it runs:

ls -s foo.txt

What -s means depends on the ls command.

Jim Ward

July 15, 2016 at 9:09 pm

Sorry, had a typo, it should be:

So when find matches foo.txt, it runs:

swathi

August 2, 2016 at 2:02 am

-s it print the allocated size

Satya

April 8, 2016 at 3:03 pm

Thank you very much. i FIND it very helpful :-) Could you also please post some info regarding finding all files that contains a particular search text.

Alan

April 6, 2016 at 1:01 am

Isn't command 25 deleting everyting larger than 10M, not 100M like in the description?

Nimant

March 8, 2016 at 10:10 pm

Thanks a lot. Clear and complete !

Efstathios

February 20, 2016 at 4:04 pm

Very good. Thank you.

Rio

January 27, 2016 at 8:08 pm

Awesome article . Great work

Bob K

June 29, 2015 at 7:07 pm

Rakesh, when you use {} ; with an exec statement the find utility will replace {} with the path and filename. In essence in the the example above if the following were the results without an exec command:

```
$ find /home/bob/dir -type f -name *.log -size +10M
/home/bob/dir/large.log
/home/bob/dir/even_larger.log
```

What find would do is run 2 separate statements:

```
$ find /home/bob/dir -type f -name *.log -size +10M -exec rm -f {} ;
rm -f /home/bob/dir/large.log
and
rm -f /home/bob/dir/even_larger.log
```

It is helpful to know that the path is relative, so if you were to say be in the /home/bob folder and use:

```
find . type f -name *.log -size +10M
```

Your results would end up like this:

```
./dir/large.log
./dir/even_larger.log
```

Wellington Torrejais

December 24, 2014 at 8:08 am

Thanks…

SR

[December 10, 2014 at 3:03 pm]

Very good briefing on FIND. perhaps AWK should be next? :)

Wolfgang

[October 12, 2014 at 2:02 pm]

I think you've got an mistake: Hidden files do NOT start with a period, they start with a dot. In your example you search in your home dir for hidden files.
The way to find executable files by usind find ist the "-executable" flag and you have to know that directories executeables, too.
If you just want to find executable files you can use

find PATH -executable -type f -name "whatever*"

on the other hand, if you looking for directories you can use

find PATH -executable -type d -name "whatever*"

> **amish**
>
> [May 6, 2015 at 5:05 pm]
>
> what's the difference between dot and period in the command line?

Imad Jundi

[September 15, 2014 at 10:10 am]

really helpfull, thank you very much

Saurav Roy

September 4, 2014 at 4:04 pm

All the commands really useful…Thank you very much

Harpal

June 4, 2014 at 6:06 pm

thanks good article

flatcap

December 7, 2013 at 2:02 am

Good examples, well explained.
But…
You need to quote all the wildcards in examples 2, 3, 4, 6, 7, 26.
Unless you're going to explain to your readers why commands might fail unexpectly :-)

Maze

August 13, 2013 at 4:04 am

Nice article, I bookmarked it for reference. It'd be interesting to have an offline version available for download, preferably as simple text, manpage or texinfo format.