find command examples

1. Find Files Using Name in Current Directory

Find all the files whose name is **tecmint.txt** in a current working directory.

```
# find . -name tecmint.txt
./tecmint.txt
```

2. Find Files Under Home Directory

Find all the files under **/home** directory with name **tecmint.txt**.

```
# find /home -name tecmint.txt
/home/tecmint.txt
```

3. Find Files Using Name and Ignoring Case

Find all the files whose name is **tecmint.txt** and contains both capital and small letters in **/home** directory.

```
# find /home -iname tecmint.txt
./tecmint.txt
./Tecmint.txt
```

4. Find Directories Using Name

Find all directories whose name is **Tecmint** in **/** directory.

```
# find / -type d -name Tecmint
/Tecmint
```

5. Find PHP Files Using Name

Find all **php** files whose name is **tecmint.php** in a current working directory.

```
# find . -type f -name tecmint.php
./tecmint.php
```

6. Find all PHP Files in Directory

Find all **php** files in a directory.

```
# find . -type f -name "*.php"
./tecmint.php
./login.php
```

```
./index.php
```

Find Files Based on their Permissions

7. Find Files With 777 Permissions

Find all the files whose permissions are **777**.

```
# find . -type f -perm 0777 -print
```

8. Find Files Without 777 Permissions

Find all the files without permission **777**.

```
# find / -type f ! -perm 777
```

9. Find SGID Files with 644 Permissions

Find all the **SGID bit** files whose permissions set to **644**.

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

10. Find Sticky Bit Files with 551 Permissions

Find all the **Sticky Bit** set files whose permission are **551**.

```
# find / -perm 1551
```

11. Find SUID Files

Find all **SUID** set files.

```
# find / -perm /u=s
```

12. Find SGID Files

Find all **SGID** set files.

```
# find / -perm /g=s
```

13. Find Read Only Files

Find all **Read Only** files.

```
# find / -perm /u=r
```

14. Find Executable Files

Find all **Executable** files.

```
# find / -perm /a=x
```

15. Find Files with 777 Permissions and Chmod to 644

Find all **777** permission files and use **chmod** command to set permissions to **644**.

```
# find / -type f -perm 0777 -print -exec chmod 644 {} \;
```

16. Find Directories with 777 Permissions and Chmod to 755

Find all **777** permission directories and use **chmod** command to set permissions to **755**.

```
# find / -type d -perm 777 -print -exec chmod 755 {} \;
```

17. Find and remove single File

To find a single file called **tecmint.txt** and remove it.

```
# find . -type f -name "tecmint.txt" -exec rm -f {} \;
```

18. Find and remove Multiple File

To find and remove multiple files such as **.mp3** or **.txt**, then use.

```
# find . -type f -name "*.txt" -exec rm -f {} \;
OR
# find . -type f -name "*.mp3" -exec rm -f {} \;
```

19. Find all Empty Files

To find all empty files under certain path.

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

20. Find all Empty Directories

To file all empty directories under certain path.

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

21. File all Hidden Files

To find all hidden files, use below command.

```
# find /tmp -type f -name ".*"
```

Search Files Based On Owners and Groups

22. Find Single File Based on User

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

```
# find / -user root -name tecmint.txt
```

23. Find all Files Based on User

To find all files that belongs to user **Tecmint** under **/home** directory.

```
# find /home -user tecmint
```

24. Find all Files Based on Group

To find all files that belongs to group **Developer** under **/home** directory.

```
# find /home -group developer
```

25. Find Particular Files of User

To find all **.txt** files of user **Tecmint** under **/home** directory.

```
# find /home -user tecmint -iname "*.txt"
```

Find Files and Directories Based on Date and Time

26. Find Last 50 Days Modified Files

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

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

27. Find Last 50 Days Accessed Files

To find all the files which are accessed **50** days back.

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

28. Find Last 50-100 Days Modified Files

To find all the files which are modified more than **50** days back and less than **100** days.

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

29. Find Changed Files in Last 1 Hour

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

```
# find / -cmin -60
```

30. Find Modified Files in Last 1 Hour

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

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

31. Find Accessed Files in Last 1 Hour

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

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

Find Files and Directories Based on Size

32. Find 50MB Files

To find all **50MB** files, use.

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

33. Find Size between 50MB – 100MB

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

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

34. Find and Delete 100MB Files

To find all **100MB** files and delete them using one single command.

```
# find / -size +100M -exec rm -rf {} \;
```

35. Find Specific Files and Delete

Find all **.mp3** files with more than **10MB** and delete them using one single command.

```
# find / -type f -name *.mp3 -size +10M -exec rm {} \;
```

That's it, We are ending this post here, In our next article we will discuss more about other Linux commands in depth with practical examples. Let us know your opinions on this article using our comment section.

Basic examples

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 example), its fine to omit
the trailing slash. However, 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 parameter) in double or
single quotes. Not doing so will seem to work sometimes and give strange
results 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 ".*"
```

Find files based on permissions

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 ...
```

Search Files Based On Owners and Groups

13. Find files belonging 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
```

Search files and directories based on size

21. Find files of given 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 | head -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 19:13 ./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 {} \;
```