In this article, let us review 15 practical examples of Linux find command that will be very useful to both newbies and experts.

First, create the following sample empty files under your home directory to try some of the find command examples mentioned below.

```
# vim create_sample_files.sh

touch MybashProgram.sh

touch mycprogram.c

touch MyCProgram.c

touch Program.c



mkdir backup

cd backup



touch MybashProgram.sh

touch mycprogram.c

touch MyCProgram.c

touch Program.c



# chmod +x create_sample_files.sh



# ./create_sample_files.sh



# ls -R

.:
```

```
backup                  MybashProgram.sh  MyCProgram.c

create_sample_files.sh  mycprogram.c      Program.c



./backup:

MybashProgram.sh  mycprogram.c  MyCProgram.c  Program.c
```

1. Find Files Using Name

This is a basic usage of the find command. This example finds all files with name — MyCProgram.c in the current directory and all its sub-directories.

```
# find -name "MyCProgram.c"

./backup/MyCProgram.c

./MyCProgram.c
```

2. Find Files Using Name and Ignoring Case

This is a basic usage of the find command. This example finds all files with name — MyCProgram.c (ignoring the case) in the current directory and all its sub-directories.

```
# find -iname "MyCProgram.c"

./mycprogram.c

./backup/mycprogram.c

./backup/MyCProgram.c

./MyCProgram.c
```

3. Limit Search To Specific Directory Level Using mindepth and maxdepth

Find the passwd file under all sub-directories starting from root directory.

```
# find / -name passwd

./usr/share/doc/nss_ldap-253/pam.d/passwd

./usr/bin/passwd

./etc/pam.d/passwd

./etc/passwd
```

Find the passwd file under root and one level down. (i.e root — level 1, and one sub-directory — level 2)

```
# find -maxdepth 2 -name passwd

./etc/passwd
```

Find the passwd file under root and two levels down. (i.e root — level 1, and two sub-directories — level 2 and 3)

```
# find / -maxdepth 3 -name passwd

./usr/bin/passwd

./etc/pam.d/passwd

./etc/passwd
```

Find the password file between sub-directory level 2 and 4.

```
# find -mindepth 3 -maxdepth 5 -name passwd

./usr/bin/passwd

./etc/pam.d/passwd
```

4. Executing Commands on the Files Found by the Find Command.

In the example below, the find command calculates the md5sum of all the files with the name MyCProgram.c (ignoring case). {} is replaced by the current file name.

```
# find -iname "MyCProgram.c" -exec md5sum {} \;

d41d8cd98f00b204e9800998ecf8427e  ./mycprogram.c

d41d8cd98f00b204e9800998ecf8427e  ./backup/mycprogram.c

d41d8cd98f00b204e9800998ecf8427e  ./backup/MyCProgram.c

d41d8cd98f00b204e9800998ecf8427e  ./MyCProgram.c
```

5. Inverting the match.

Shows the files or directories whose name are not MyCProgram.c .Since the maxdepth is 1, this will look only under current directory.

```
# find -maxdepth 1 -not -iname "MyCProgram.c"

.

./MybashProgram.sh

./create_sample_files.sh

./backup

./Program.c
```

6. Finding Files by its inode Number.

Every file has an unique inode number, using that we can identify that file. Create two files with similar name. i.e one file with a space at the end.

```
# touch "test-file-name"



# touch "test-file-name "

[Note: There is a space at the end]
```

```
# ls -1 test*

test-file-name

test-file-name
```

From the ls output, you cannot identify which file has the space at the end. Using option -i, you can view the inode number of the file, which will be different for these two files.

```
# ls -i1 test*

16187429 test-file-name

16187430 test-file-name
```

You can specify inode number on a find command as shown below. In this example, find command renames a file using the inode number.

```
# find -inum 16187430 -exec mv {} new-test-file-name \;



# ls -i1 *test*

16187430 new-test-file-name

16187429 test-file-name
```

You can use this technique when you want to do some operation with the files which are named poorly as shown in the example below. For example, the file with name — file?.txt has a special character in it. If you try to execute “rm file?.txt”, all the following three files will get removed. So, follow the steps below to delete only the “file?.txt” file.

```
# ls

file1.txt  file2.txt  file?.txt
```

Find the inode numbers of each file.

```
# ls -i1

804178 file1.txt

804179 file2.txt

804180 file?.txt
```

Use the inode number to remove the file that had special character in it as shown below.

```
# find -inum 804180 -exec rm {} \;



# ls

file1.txt  file2.txt

[Note: The file with name "file?.txt" is now removed]
```

7. Find file based on the File-Permissions

Following operations are possible.

- Find files that match exact permission
- Check whether the given permission matches, irrespective of other permission bits
- Search by giving octal / symbolic representation

For this example, let us assume that the directory contains the following files. Please note that the file-permissions on these files are different.

```
# ls -l

total 0

-rwxrwxrwx 1 root root 0 2009-02-19 20:31 all_for_all

-rw-r--r-- 1 root root 0 2009-02-19 20:30 everybody_read

---------- 1 root root 0 2009-02-19 20:31 no_for_all
```

```
-rw------- 1 root root 0 2009-02-19 20:29 ordinary_file

-rw-r----- 1 root root 0 2009-02-19 20:27 others_can_also_read

----r----- 1 root root 0 2009-02-19 20:27 others_can_only_read
```

Find files which has read permission to group. Use the following command to find all files that are readable by the world in your home directory, irrespective of other permissions for that file.

```
# find . -perm -g=r -type f -exec ls -l {} \;

-rw-r--r-- 1 root root 0 2009-02-19 20:30 ./everybody_read

-rwxrwxrwx 1 root root 0 2009-02-19 20:31 ./all_for_all

----r----- 1 root root 0 2009-02-19 20:27 ./others_can_only_read

-rw-r----- 1 root root 0 2009-02-19 20:27 ./others_can_also_read
```

Find files which has read permission only to group.

```
# find . -perm g=r -type f -exec ls -l {} \;

----r----- 1 root root 0 2009-02-19 20:27 ./others_can_only_read
```

Find files which has read permission only to group [search by octal]

```
# find . -perm 040 -type f -exec ls -l {} \;

----r----- 1 root root 0 2009-02-19 20:27 ./others_can_only_read
```

8. Find all empty files (zero byte file) in your home directory and its subdirectory

Most files of the following command output will be lock-files and place holders created by other applications.

```
# find ~ -empty
```

List all the empty files only in your home directory.

```
# find . -maxdepth 1 -empty
```

List only the non-hidden empty files only in the current directory.

```
# find . -maxdepth 1 -empty -not -name ".*"
```

9. Finding the Top 5 Big Files

The following command will display the top 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
```

10. Finding the Top 5 Small Files

Technique is same as finding the bigger files, but the only difference the sort is ascending order.

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

In the above command, most probably you will get to see only the ZERO byte files (empty files). So, you can use the following command to list the smaller files other than the ZERO byte files.

```
# find . -not -empty -type f -exec ls -s {} \; | sort -n  | head -5
```

11. Find Files Based on file-type using option -type

Find only the socket files.

```
# find . -type s
```

Find all directories

```
# find . -type d
```

Find only the normal files

```
# find . -type f
```

Find all the hidden files

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

Find all the hidden directories

```
# find -type d -name ".*"
```

12. Find files by comparing with the modification time of other file.

Show files which are modified after the specified file. The following find command displays all the files that are created/modified after ordinary_file.

```
# ls -lrt

total 0

-rw-r----- 1 root root 0 2009-02-19 20:27 others_can_also_read

----r----- 1 root root 0 2009-02-19 20:27 others_can_only_read

-rw------- 1 root root 0 2009-02-19 20:29 ordinary_file

-rw-r--r-- 1 root root 0 2009-02-19 20:30 everybody_read

-rwxrwxrwx 1 root root 0 2009-02-19 20:31 all_for_all

---------- 1 root root 0 2009-02-19 20:31 no_for_all



# find -newer ordinary_file
```

```
.

./everybody_read

./all_for_all

./no_for_all
```

13. Find Files by Size

Using the -size option you can find files by size.

Find files bigger than the given size

```
# find ~ -size +100M
```

Find files smaller than the given size

```
# find ~ -size -100M
```

Find files that matches the exact given size

```
# find ~ -size 100M
```

Note: – means less than the give size, + means more than the given size, and no symbol means exact given size.

14. Create Alias for Frequent Find Operations

If you find some thing as pretty useful, then you can make it as an alias. And execute it whenever you want.

Remove the files named a.out frequently.

```
# alias rmao="find . -iname a.out -exec rm {} \;"

# rmao
```

Remove the core files generated by c program.

```
# alias rmc="find . -iname core -exec rm {} \;"

# rmc
```

15. Remove big archive files using find command

The following command removes *.zip files that are over 100M.

```
# find / -type f -name *.zip -size +100M -exec rm -i {} \;"
```

Remove all *.tar file that are over 100M using the alias rm100m (Remove 100M). Use the similar concepts and create alias like rm1g, rm2g, rm5g to remove file size greater than 1G, 2G and 5G respectively.

```
# alias rm100m="find / -type f -name *.tar -size +100M -exec rm -i {} \;"

# alias rm1g="find / -type f -name *.tar -size +1G -exec rm -i {} \;"

# alias rm2g="find / -type f -name *.tar -size +2G -exec rm -i {} \;"

# alias rm5g="find / -type f -name *.tar -size +5G -exec rm -i {} \;"



# rm100m

# rm1g

# rm2g

# rm5g
```