Mommy, I found it! — 15 Practical Linux Find Command Examples

Photo courtesy of [Qole Pejorian](http://www.flickr.com/photos/qole/792291810/)

Apart from the basic operation of looking for files under a directory structure, you can also perform several practical operations using find command that will make your command line journey easy.  
  
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

### Find Command Examples Second Part

If you liked this Mommy article on find command, don’t forget to check-out the Daddy article of the find command — [Daddy, I found it!, 15 Awesome Linux Find Command Examples (Part2)](http://www.thegeekstuff.com/2009/06/15-practical-unix-linux-find-command-examples-part-2/)

# Daddy, I found it!, 15 Awesome Linux Find Command Examples (Part2)

A while back we reviewed 15 practical [find command examples (Part I)](http://www.thegeekstuff.com/2009/03/15-practical-linux-find-command-examples/). Find command can do lot more than just searching for files based on name.  
   
In this article (Part 2), let us discuss 15 advanced examples of **find command** including — finding files based on the time it is accessed, modified or changed, finding files comparatively, performing operation on found files etc.,  
   
**Ramesh Natarajan:** That is my sweet little daughter in that picture. She was very happy to spot the sea lion in the California Long Beach Aquarium.

## Find Files Based on Access / Modification / Change Time

You can find files based on following three file time attribute.

1. **Access time** of the file. Access time gets updated when the **file accessed**.
2. **Modification time** of the file. Modification time gets updated when the **file content modified**.
3. **Change time** of the file. Change time gets updated when the **inode data changes**.

In the following examples, the difference between the **min option** and the **time option** is the argument.

* **min argument** treats its argument as **minutes**. For example, min 60 = 60 minutes (1 hour).
* **time argument** treats its argument as **24 hours**. For example, time 2 = 2\*24 hours (2 days).
* While doing the 24 hours calculation, the fractional parts are ignored so 25 hours is taken as 24 hours, and 47 hours is also taken as 24 hours, only 48 hours is taken as 48 hours. To get more clarity refer the -atime section of the**find command** man page.

### Example 1: Find files whose content got updated within last 1 hour

To find the files based up on the content modification time, the option -mmin, and -mtime is used. Following is the definition of mmin and mtime from man page.

* **-mmin n** File’s data was last modified **n minutes** ago.
* **-mtime n** File’s data was last modified **n\*24 hours** ago.

Following example will find files in the current directory and sub-directories, whose content got updated within last 1 hour (60 minutes)

# find . -mmin -60

In the same way, following example finds all the files (under root file system /) that got updated within the last 24 hours (1 day).

# find / -mtime -1

### Example 2: Find files which got accessed before 1 hour

To find the files based up on the file access time, the option -amin, and -atime is used. Following is the definition of amin and atime from find man page.

* **-amin n** File was last accessed **n minutes** ago
* **-atime n** File was last accessed **n\*24 hours** ago

Following example will find files in the current directory and sub-directories, which got accessed within last 1 hour (60 minutes)

# find -amin -60

In the same way, following example finds all the files (under root file system /) that got accessed within the last 24 hours (1 day).

# find / -atime -1

### Example 3: Find files which got changed exactly before 1 hour

To find the files based up on the file inode change time, the option -cmin, and -ctime is used. Following is the definition of cmin and ctime from find man page.

* **-cmin n** File’s status was last changed **n minutes** ago.
* **-ctime n** File’s status was last changed **n\*24 hours** ago.

Following example will find files in the current directory and sub-directories, which changed within last 1 hour (60 minutes)

# find . -cmin -60

In the same way, following example finds all the files (under root file system /) that got changed within the last 24 hours (1 day).

# find / -ctime -1

### Example 4: Restricting the find output only to files. (Display only files as find command results)

The above find command’s will also show the directories because directories gets accessed when the file inside it gets accessed. But if you want only the files to be displayed then give -type f in the find command as  
   
The following **find command** displays files that are accessed in the last 30 minutes.

# find /etc/sysconfig -amin -30

.

./console

./network-scripts

./i18n

./rhn

./rhn/clientCaps.d

./networking

./networking/profiles

./networking/profiles/default

./networking/profiles/default/resolv.conf

./networking/profiles/default/hosts

./networking/devices

./apm-scripts

[Note: The above output contains both files and directories]

# find /etc/sysconfig -amin -30 -type f

./i18n

./networking/profiles/default/resolv.conf

./networking/profiles/default/hosts

[Note: The above output contains only files]

### Example 5: Restricting the search only to unhidden files. (Do not display hidden files in find output)

When we don’t want the hidden files to be listed in the find output, we can use the following regex.  
The below find displays the files which are modified in the last 15 minutes. And it lists only the unhidden files. i.e hidden files that starts with a . (period) are not displayed in the find output.

# find . -mmin -15 \( ! -regex ".\*/\..\*" \)

## Finding Files Comparatively Using Find Command

Human mind can remember things better by reference such as, i want to find files which i edited after editing the file “test”. You can find files by referring to the other files modification as like the following.

### Example 6: Find files which are modified after modification of a particular FILE

Syntax: find -newer FILE

Following example displays all the files which are modified after the /etc/passwd files was modified. This is helpful, if you want to track all the activities you’ve done after adding a new user.

# find -newer /etc/passwd

### Example 7: Find files which are accessed after modification of a specific FILE

Syntax: find -anewer FILE

Following example displays all the files which are accessed after modifying /etc/hosts. If you remember adding an entry to the /etc/hosts and would like to see all the files that you’ve accessed since then, use the following command.

# find -anewer /etc/hosts

### Example 8: Find files whose status got changed after the modification of a specific FILE.

Syntax: find -cnewer FILE

Following example displays all the files whose status got changed after modifying the /etc/fstab. If you remember adding a mount point in the /etc/fstab and would like to know all the files who status got changed since then, use the following command.

find -cnewer /etc/fstab

## Perform Any Operation on Files Found From Find Command

We have looked at many different ways of finding files using **find command** in this article and also in our previous article. If you are not familiar in finding files in different ways, i strongly recommend you to read the part 1.  
   
This section explains about how to do different operation on the files from the find command. i.e how to manipulate the files returned by the find command output.  
   
We can specify any operation on the files found from find command.

find <CONDITION to Find files> -exec <OPERATION> \;

The OPERATION can be anything such as:

* **rm command** to remove the files found by find command.
* **mv command** to rename the files found.
* **ls -l command** to get details of the find command output files.
* **md5sum** on find command output files
* **wc command** to count the total number of words on find command output files.
* Execute any **Unix shell command** on find command output files.
* or Execute your own **custom shell script** / command on find command output files.

### Example 9: ls -l in find command output. Long list the files which are edited within the last 1 hour.

# find -mmin -60

./cron

./secure

# find -mmin -60 -exec ls -l {} \;

-rw------- 1 root root 1028 Jun 21 15:01 ./cron

-rw------- 1 root root 831752 Jun 21 15:42 ./secure

### Example 10: Searching Only in the Current Filesystem

System administrators would want to search in the root file system, but not in the other mounted partitions. When you have multiple partitions mounted, and if you want to search in /. You can do the following.  
   
Following command will search for \*.log files starting from /. i.e If you have multiple partitions mounted under / (root), the following command will search all those mounted partitions.

# find / -name "\*.log"

This will search for the file only in the current file system. Following is the xdev definition from find man page:

* **-xdev** Don’t descend directories on other filesystems.

Following command will search for \*.log files starting from / (root) and only in the current file system. i.e If you have multiple partitions mounted under / (root), the following command will NOT search all those mounted partitions.

# find / -xdev -name "\*.log"

### Example 11: Using more than one { } in same command

Manual says only one instance of the {} is possible. But you can use more than one {} in the same command as shown below.

# find -name "\*.txt" cp {} {}.bkup \;

Using this {} in the same command is possible but using it in different command it is not possible, say you want to rename the files as following, which will not give the expected result.

find -name "\*.txt" -exec mv {} `basename {} .htm`.html \;

### Example 12: Using { } in more than one instance.

You can simulate it by writing a shell script as shown below.

# mv "$1" "`basename "$1" .htm`.html"

These double quotes are to handle spaces in file name. And then call that shell script from the **find command** as shown below.

find -name "\*.html" -exec ./mv.sh '{}' \;

So for any reason if you want the same file name to be used more than once then writing the simple shell script and passing the file names as argument is the simplest way to do it.

### Example 13: Redirecting errors to /dev/null

Redirecting the errors is not a good practice. An experienced user understands the importance of getting the error printed on terminal and fix it.  
   
Particularly in find command redirecting the errors is not a good practice. But if you don’t want to see the errors and would like to redirect it to null do it as shown below.

find -name "\*.txt" 2>>/dev/null

Sometimes this may be helpful. For example, if you are trying to find all the \*.conf file under / (root) from your account, you may get lot of “Permission denied” error message as shown below.

$ find / -name "\*.conf"

/sbin/generate-modprobe.conf

find: /tmp/orbit-root: Permission denied

find: /tmp/ssh-gccBMp5019: Permission denied

find: /tmp/keyring-5iqiGo: Permission denied

find: /var/log/httpd: Permission denied

find: /var/log/ppp: Permission denied

/boot/grub/grub.conf

find: /var/log/audit: Permission denied

find: /var/log/squid: Permission denied

find: /var/log/samba: Permission denied

find: /var/cache/alchemist/printconf.rpm/wm: Permission denied

[Note: There are two valid \*.conf files burned in the "Permission denied" messages]

So, if you want to just view the real output of the **find command** and not the “Permission denied” error message you can redirect the error message to /dev/null as shown below.

$ find / -name "\*.conf" 2>>/dev/null

/sbin/generate-modprobe.conf

/boot/grub/grub.conf

[Note: All the "Permission denied" messages are not displayed]

### Example 14: Substitute space with underscore in the file name.

Audio files you download from internet mostly come with the spaces in it. But having space in the file name is not so good for Linux kind of systems. You can use the find and rename command combination as shown below to rename the files, by substituting the space with underscore.  
   
The following replaces space in all the \*.mp3 files with \_

$ find . -type f -iname “\*.mp3″ -exec rename “s/ /\_/g” {} \;

### Example 15: Executing two find commands at the same time

As shown in the examples of the find command in its manual page, the following is the syntax which can be used to execute two commands in single traversal.  
   
The following find command example, traverse the filesystem just once, listing setuid files and directories into /root/suid.txt and large files into /root/big.txt.

# find / \( -perm -4000 -fprintf /root/suid.txt '%#m %u %p\n' \) , \

\( -size +100M -fprintf /root/big.txt '%-10s %p\n' \)