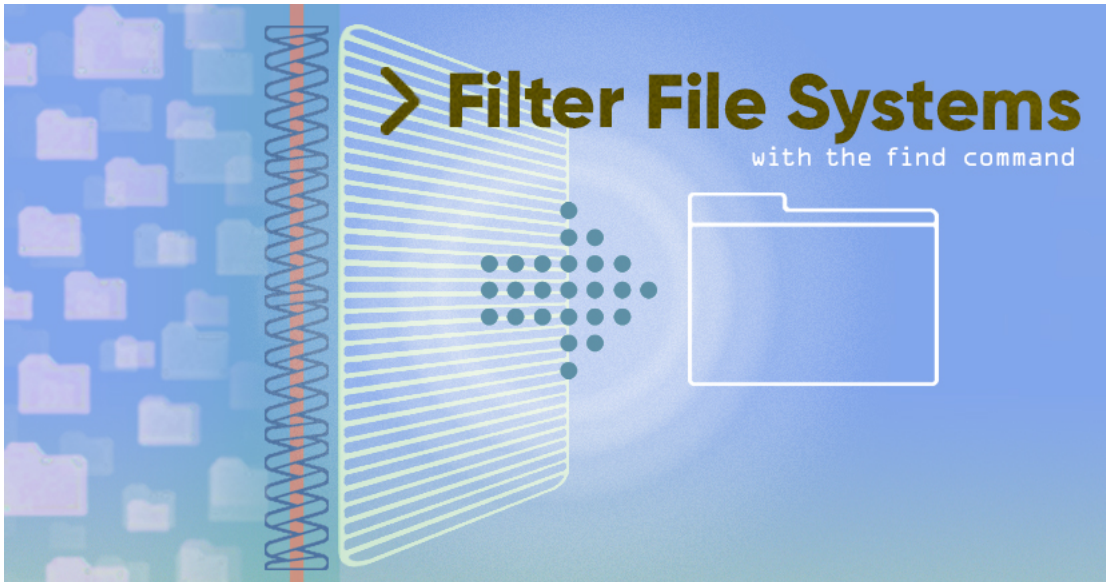
Find Files in Linux, Using the Command Line

Updated Thursday, September 15, 2016 by Edward Angert

Written by Linode

`find` is a command for recursively filtering objects in the file system based on a simple conditional mechanism. Use `find` to search for a file or directory on your file system. Using the `-exec` flag, files can be found and immediately processed within the same command.



Find Linux Files by Name or Extension

Use `find` from the command line to locate a specific file by name or extension. The following example searches for `*.err` files in the `/home/username/` directory and all sub-directories:

```
find /home/username/ -name "*.err"
```

Common Linux Find Commands and Syntax

`find` expressions take the following form:

```
find options starting/path expression
```

- The `options` attribute will control the behavior and optimization method of the `find` process.
- The `starting/path` attribute will define the top level directory where `find` begins filtering.

- The `expression` attribute controls the tests that search the directory hierarchy to produce output.

Consider the following example command:

```
find -O3 -L /var/www/ -name "*.html"
```

This command enables the maximum optimization level (-O3) and allows `find` to follow symbolic links (`-L`). `find` searches the entire directory tree beneath `/var/www/` for files that end with `.html` .

Basic Examples

Command	Description
`find . -name testfile.txt`	Find a file called testfile.txt in current and sub-directories.
`find /home -name *.jpg`	Find all `.jpg` files in the `/home` and sub-directories.
`find . -type f -empty`	Find an empty file within the current directory.
`find /home -user exampleuser -mtime 7 -iname ".db"`	Find all `.db` files (ignoring text case) modified in the last 7 days by a user named exampleuser.

Options and Optimization for Find

The default configuration for `find` will ignore symbolic links (shortcut files). If you want `find` to follow and return symbolic links, you can add the `-L` option to the command, as shown in the example above.

`find` optimizes its filtering strategy to increase performance. Three user-selectable optimization levels are specified as `-O1` , `-O2` , and `-O3` . The `-O1` optimization is the default and forces `find` to filter based on filename before running all other tests.

Optimization at the `-O2` level prioritizes file name filters, as in `-O1` , and then runs all file-type filtering before proceeding with other more resource-intensive conditions. Level `-O3` optimization allows `find` to perform the most severe optimization and reorders all tests based on their relative expense and the likelihood of their success.

Command	Description
`-O1`	(Default) filter based on file name first.
`-O2`	File name first, then file-type.
`-O3`	Allow `find` to automatically re-order the search based on efficient use of resources and likelihood. of success
`-maxdepth X`	Search current directory as well as all sub-directories X levels deep.
`-iname`	Search without regard for text case.
`-not`	Return only results that do not match the test case.
`-type f`	Search for files.
`-type d`	Search for directories.

Find Files by Modification Time

The `find` command contains the ability to filter a directory hierarchy based on when the file was last modified:

```
find / -name "*conf" -mtime 7
find /home/exampleuser/ -name "*conf" -mtime 3
```

The first command returns a list of all files in the entire file system that end with the characters `conf` and have been modified in the last 7 days. The second command filters `exampleuser` user's home directory for files with names that end with the characters `conf` and have been modified in the previous 3 days.

Use Grep to Find Files Based on Content

The `find` command is only able to filter the directory hierarchy based on a file's name and meta data. If you need to search based on the content of the file, use a tool like grep (/docs/tools-reference/search-and-filter-text-with-grep). Consider the following example:

```
find . -type f -exec grep "example" '{}' \; -print
```

This searches every object in the current directory hierarchy (`.`) that is a file (`-type f`) and then runs the command `grep "example"` for every file that satisfies the conditions. The files that match are printed on the screen (`-print`). The curly braces (`{}`) are a placeholder for the `find` match results. The `{}` are enclosed in single quotes (`'`) to avoid handing `grep` a malformed file name. The `-exec` command is terminated with a semicolon (`;`), which should be escaped (`\;`) to avoid interpretation by the shell.

Before the implementation of the `-exec` option, this kind of command might have used the `xargs` command to generate a similar output:

```
find . -type f -print | xargs grep "example"
```

How to Find and Process Files Using the Find Command

The `-exec` option runs commands against every object that matches the find expression. Consider the following example:

```
find . -name "rc.conf" -exec chmod o+r '{}' \;
```

This filters every object in the current hierarchy (`.`) for files named `rc.conf` and runs the `chmod o+r` command to modify file permissions of the `find` results.

The commands run with the `-exec` are executed in the root directory of the `find` process. Use `-execdir` to execute the specified command in the directory where the match resides. This may alleviate security concerns and produce more desirable performance for some operations.

The `-exec` or `-execdir` options run without further prompts. If you prefer to be prompted before action is taken, replace `-exec` with `-ok` or `-execdir` with `-okdir` .

How to Find and Delete Files in the Linux Command Line

Caution
Use this option with extreme caution.

Add the option `-delete` to the end of a match expression to delete all files that match. Use this option when you are certain that the results *only* match the files that you wish to delete.

In the following example, `find` locates all files in the hierarchy starting at the current directory and fully recursing into the directory tree. In this example, `find` will delete all files that end with the characters `.bak` :

```
find . -name "*.bak" -delete
```