**How To Find a File In Linux From the Command Line**

Minutes

Need to know how to **find a file in Linux**? Well, surprise, surprise, you’re going to need the *find* command in Linux to scour your directory or file system. The Linux *find* command can filter objects recursively using a simple conditional mechanism, and if you use the *-exec* flag, you’ll also be able to find a file in Linux straightaway and process it without needing to use another command.

Locate Linux Files by Their Name or Extension

Type *find* into the command line to track down a particular file by its name or extension. If you want to look for \*.err files in the /home/username/ directory and all sub-directories, try this:

*find* /home/username/ -name "\*.err"

Typical Linux *Find* Commands and Syntax

*find* command expressions look like this:

*find* command options *starting/path* expression

The options attribute controls the behavior and optimization method of the *find* process. The *starting/path* attribute defines the top-level directory where the *find* command in Linux begins the filtering process. The *expression* attribute controls the assessments that scour the directory tree to create output.

Let’s break down a Linux find command where we don’t just want Linux find file by name:

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

It enables the top-level optimization (-O3) and permits *find* to follow symbolic links (-L). The *find* command in Linux searches through the whole directory hierarchy under /var/www/ for files that have .html on the end.

Basic Examples

1. *find* . -name thisfile.txt

If you need to know how to find a file in Linux called thisfile.txt, it will look for it in current and sub-directories.

2. *find* /home -name \*.jpg

Look for all .jpg files in the /home and directories below it.

3. *find* . -type f -empty

Look for an empty file inside the current directory.

4. *find* /home -user randomperson-mtime 6 -iname ".db"

Look for all .db files (ignoring text case) that have been changed in the preceding 6 days by a user called *randomperson*.

Options and Optimization for *Find*Command for Linux

*find* is configured to ignore symbolic links (shortcut files) by default. If you’d like the *find* command to follow and show symbolic links, just add the -L option to the command, as we did in this example.

*find* can help Linux find file by name. The Linux*find* command enhances its approach to filtering so that performance is optimised. The user can find a file in Linux by selecting three stages of optimisation*-O1, -O2,* and *-O3. -O1* is the standard setting and it causes *find* to filter according to filename before it runs any other tests.

-O2 filters by name and type of file before carrying on with more demanding filters to find a file in Linux. Level -O3 reorders all tests according to their relative expense and how likely they are to succeed.

* -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 this directory along with all sub-directories to a level of X
* -iname – Search while ignoring text case.
* -not – Only produce results that don’t match the test case
* -type f – Look for files
* -type d – Look for directories

*Find* Files by When They Were Modified

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

find / -name "\*jpg" -mtime 5

find /home/randomuser/ -name "\*jpg" -mtime 4

The initial Linux *find* command pulls up a list of files in the whole system that end with the characters *jpg* and have been modified in the preceding 5 days. The next one filters *randomuser’s* home directory for files with names that end with the characters “conf” and have been modified in the preceding 4 days.

Use Grep to Find Files Based on Content

The *find*command in Linux is great but it can only filter the directory tree according to filename and meta data. To search files based on what they contain you’ll need a tool like grep. Take a look:

*find* . -type f *-exec* grep "forinstance" '{}' \; -print

This goes through every object in the current directory tree (.) that’s a file (-type f) and then runs grep ” forinstance ” for every file that matches, then prints them on the screen (-print). The curly braces ({}) are a placeholder for those results matched by the Linux *find*command. The {} go inside single quotes (‘) so that grep isn’t given a misshapen file name. The *-exec* command is ended with a semicolon (;), which also needs an escape (\;) so that it doesn’t end up being interpreted by the shell.

Before *-exec* was implemented, xargs would have been used to create the same kind of output:

*find* . -type f -print | xargs grep "forinstance"

How to Locate and Process Files Using the *Find* Command in Linux

The *-exec* option runs commands against every object that matches the *find* expression. Let’s see how that looks:

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

This filters all objects in the current directory tree (.) for files named rc.conf and runs the chmod o+r command to alter file permissions of the results that*find*returns.

The root directory of the Linux is where the commands that *-exec* runs are executed. Use *-execdir* to execute the command you want in the directory where the match is sitting, because this might be more secure and improve performance under certain circumstances.