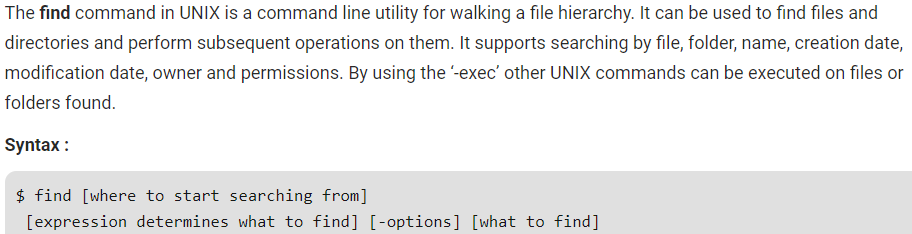
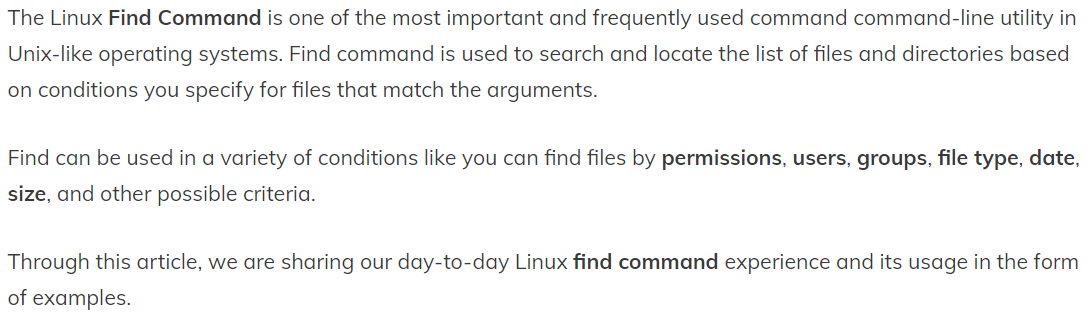
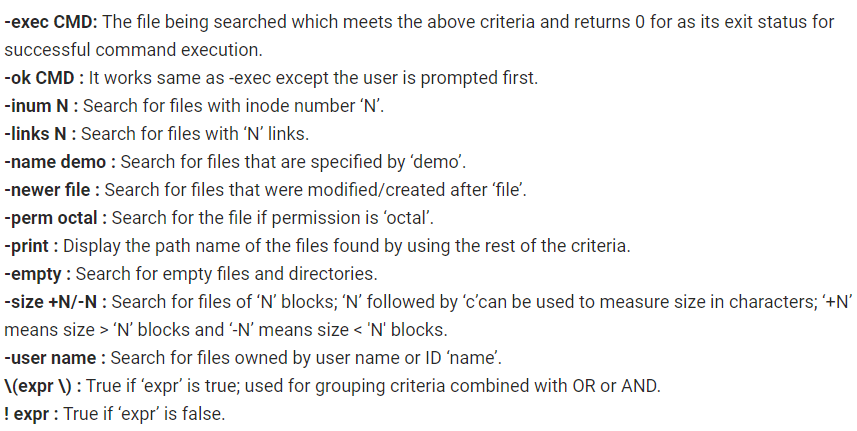
**find command:**





**Options:**



**Examples:**

Display the file in current directory with a name. we can also give the path if we want.

* **find . -name tecmint.txt**

finding the files ignore the case. These same commands can also be used to find the directories.

* **find . -iname tecmint.txt**

finding only files but not directories.

* **find . -type f -name tecmint.php**

finding only directories.

* **find / -type d -name Tecmint**

finding files or directories with wildcard entries.

* **find . -type f -name "\*.php"**
* **find . -type d -name "\*php"**

finding files with permissions.

* **find . -type f -perm 0777 -print**

finding files without permissions.

* **find / -type f ! -perm 777**

Same as the above command but uses a symbolic representation of the permission bits. Note that the symbolic notation uses a comma separator and contains no spaces.

* **find . -perm u=rwx,g=rx,o=r**

Locate and print a list of any file or directory in or below the current directory owned by the user jeff.

* **find . -user jeff**

Locate and print a list of any file in or below the current directory whose owning group is the dev group.

* **find . -group dev**

Search for empty files and directories. This command finds all empty folders and files in the entered directory or sub-directories.

* **find ./GFG -empty**
* **find /tmp -type f -empty**
* **find /tmp -type d -empty**

find files and change the permissions.

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

find a file and remove them.

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

find file modified 50 days back.

* **find / -mtime 50**

filnd files which are accessed 50days back.

* **find / -atime 50**

find files which are modified more than 50 days and less than 100 days.

* **find / -mtime +50 –mtime -100**

find files which are modified, accessed, changes in last 1 hour.

* **find / -cmin -60**
* **find / -mmin -60**
* **find / -amin -60**

find files which are 50 MB size.

* **find / -size 50M**

find files which are greater then 50 MB and less than 100 MB

* **find / -size +50M -size -100M**

Locate and print a list of any regular file in or below the current directory whose size is less than 5 gigabytes.

* **find . -size -5G -type f**

find all mp3 files which are more than 10 MB and delete them.

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

Locate any files in or below the current directory whose size is greater than 1 gigabyte, and execute the mv command on them, moving them into the directory bigfiles in your home directory. The {} indicates where in the command the name of the matched file should be placed; it must be enclosed in quotes to protect it from being misinterpreted by the shell. Similarly, the semicolon which ends the command must be escaped with a backslash ("\;").

* **find . -size +1G -exec mv '{}' ~/bigfiles \;**

Using -ok is the same as using -exec, but you will be asked for confirmation before each command is executed.

* **find . -size +1G -ok mv '{}' ~/bigfiles \;**

Locate any files in or below the current directory whose suffix is ".jpg" or ".gif". The -o option functions as a boolean OR operator; if either of the conditions are true, the file will be included in the list.

* **find . -name '\*.jpg' -o -name '\*.gif'**

Locate any files in or below the current directory whose suffix is ".jpg", but limit subdirectory traversal to 2 levels beneath the current directory. Any subdirectories 3 or more levels deep will not be searched.

* **find . -maxdepth 2 -name '\*.jpg'**

Locate any files in or below the current directory whose suffix is not ".jpg". The exclamation mark ("!") functions as a boolean NOT operator; it lists only files for which the condition is false.

* **find . ! -name '\*.jpg'**

Find files named core in or below the directory /tmp and delete them. The -name test comes before the -type test to avoid having to call stat on every file.

* **find /tmp -name core -type f -print0 | xargs -0 /bin/rm -f**

Search for files in your home directory which have been modified in the last twenty-four hours. This command works this way because the time since each file was last modified is divided by 24 hours and any remainder is discarded. That means that to match -mtime 0, a file will have to have a modification in the past that is less than 24 hours ago.

* **find $HOME -mtime 0**

Search for files in your superuser binary directories, /sbin and /usr/sbin, which are executable but not readable

* **find /sbin /usr/sbin -executable ! -readable**

Here, 2> is a special operator in bash which means "redirect standard error". This will effectively hide all error messages from find's output.

You might not want to hide every error message, though. What if there's another error of some kind? You don't want to suppress those messages too. In that case, you can use this command instead:

* **find . 2>/dev/null**

Here, 2> tells bash to redirect standard error, and &1 tells it to use standard output as the destination. This redirects all error messages issued by find to standard output; this will look the same on your terminal screen, but by merging them with standard output, we have made them filterable by grep. So we then pipe the output of find to grep, which matches the inverse (-v) of our string, 'Permission denied'. The result is that it displays any line which does not contain that string. This allows you to view your results without any pesky "Permission denied" error messages.

* **find . 2>&1 | grep -v 'Permission denied'**

Locate any files in or below the current directory whose size is greater than 1 gigabyte, then pipe that list to the xargs command, which uses the mv command to move each one of those files into the directory bigfiles in your home directory. This is similar to the above command, but better for several reasons. First, it uses the -print0 option to tell find to create its list separating each file name with a null character rather than a newline; this makes the list difficult for a human to read, but has the advantage of making it easier for another program to parse. You should always use -print0 when piping output to xargs.

Using xargs to execute commands on every file found is generally better than using find's -exec option because of the more efficient way xargs threads each command that it spawns.

The -0 argument to xargs tells it to expect the null character as the file name separator (which we specified with find's -print0 option).

The -I '{}' option tells xargs to replace "{}" with the name of each file it finds. We then form our command using {} where we want the file name to appear. We enclose it in single quotes to protect it from the shell.

xargs does not expect a semicolon at the end of the command, unlike find -exec, so it is not included in this command.

* **find . -size +1G -print0 | xargs -0 -I '{}' mv '{}' ~/bigfiles**