The FIND command is a powerful and flexible way to locate data on your system.

Find can be used in variety of conditions like you can find files by permissions, users, groups, file type, date, size and other possible criteria.

The find command is one of the most essential commands on the linux terminal, that enables searching of files very easy.

- Find (and print) every file on the system.

```
$ find /
```

- The -name argument allows you to restrict your results to files that match the given pattern.The tilde (~) is a metacharacter commonly used for denoting the home folder of the current user.

```
$ find ~ -name '*jpg'
```

- What if some of them have an uppercase extension? -iname is like -name, but it is case-insensitive.

```
$ find ~ -iname '*jpg'
```

- Find a file "foo.bar" that exists somewhere in the filesystem

```
$ find / -name foo.bar -print
```

- Find Files Using Name in Current Directory

```
# find . -name tecmint.txt
```

The first part of the find command is obviously the word find.
The second part is where to start searching from.
The next part is an expression which determines what to find.
Finally the last part is the name of the thing to find.

- Find all the files under /home directory with name tecmint.txt.

```
# find /home -name tecmint.txt
```

- Find all the files whose name is tecmint.txt and contains both capital and small letters in /home directory.

```
# find /home -iname tecmint.txt
```

- Find all directories whose name is Tecmint in / directory.

```
# find / -type d -name Tecmint
```

- Find all php files whose name is tecmint.php in a current working directory.

```
# find . -type f -name tecmint.php
```

- Find all php files in a directory.

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

- Find a file without showing "Permission Denied" messages

```
$ find / -name foo.bar -print 2>/dev/null
```

The 2>/dev/null option sends these messages to /dev/null so that the found files are easily viewed.

- Find a file without searching network or mounted filesystems

```
$ find / -name foo.bar -print -xdev
```

- Find a file, who's name ends with .bar, within the current directory and only search 2 directories deep

```
$ find . -name *.bar -maxdepth 2
```

- Search directories "./dir1" and "./dir2" for a file "foo.bar"

```
$ find ./dir1 ./dir2 -name foo.bar
```

- Search for files that are owned by the user "joebob"

```
$ find /some/directory -user joebob
```

- Find a file that is a certain type. "-type l" searches for symbolic links

```
$ find /some/directory -type l
```

Several types of files can be searched for:

```
b       block (buffered) special
c       character (unbuffered) special
d       directory
p       named pipe (FIFO)
f        regular file
l        symbolic link
s       socket
D     door (Solaris)
```

- Search for directories that contain the phrase "foo" but do not end in ".bar"

```
$ find . -name '*foo*' ! -name '*.bar' -type d
```

The "!" allows you to exclude results that contain the phrases following it.

- **The power of find**

find becomes extremely useful when combined with other commands. One such combination would be using find and grep together.

```
$ find ~/documents -type f -name '*.txt' \
-exec grep -s DOGS {} \;
```

This sequence uses find to look in /users/home/directory/documents for a file (-type f) with a name ending in .txt. It sends the files it finds to the grep command via the -exec option and grep searches the file found for any occurrences of the word "DOG". If the file is found it will be output to the screen and if the word "DOG" is found, within one of the found files, the line that "DOG" occurs in will also be output to the screen.

- Some of the pictures might have a .jpeg extension. Fortunately, we can combine patterns with an "or," represented by -o.

```
$ find ~ ( -iname 'jpeg' -o -iname 'jpg' )
```

- What if you have some directories that end in jpg? (Why you named a directory bucketofjpg instead of pictures is beyond me.) We can modify our command with the -type argument to look only for files.

```
$ find ~ \( -iname '*jpeg' -o -iname '*jpg' \) -type f
```

- Or maybe you'd like to find those oddly named directories so you can rename them later:

```
$ find ~ \( -iname '*jpeg' -o -iname '*jpg' \) -type d
```

- It turns out you've been taking a lot of pictures lately, so let's narrow this down to files that have changed in the last week.

```
$ find ~ \( -iname '*jpeg' -o -iname '*jpg' \) -type f -mtime -7
```

You can do time filters based on file status change time (ctime), modification time (mtime), or access time (atime). These are in days, so if you want finer-grained control, you can express it in minutes instead (cmin, mmin, and amin, respectively). Unless you know exactly the time you want, you'll probably prefix the number with + (more than) or – (less than).

- Maybe you're running out of disk space, so you want to find all the gigantic (let's define that as "greater than 1 gigabyte") files in the log directory:

```
$find /var/log -size +1G
```

- Or maybe you want to find all the files owned by bcotton in /data:

```
$find /data -owner bcotton
```

There are other expressions you can use as follows:

- -amin n - The file was last accessed n minutes ago
- -anewer - The file was last accessed more recently than it was modified
- -atime n - The file was last accessed more n days ago
- -cmin n - The file was last changed n minutes ago
- -cnewer - The file was last changed more recently than the file was modified
- -ctime n - The file was last changed more than n days ago
- -empty - The file is empty
- -executable - The file is executable
- -false - Always false
- -fstype type - The file is on the specified file system
- -gid n - The file belongs to group with the ID n
- -group groupname - The file belongs to the named group
- -ilname pattern - Search for a symbolic line but ignore case
- -iname pattern - Search for a file but ignore case
- -inum n - Search for a file with the specified node
- -ipath path - Search for a path but ignore case
- -iregex expression - Search for a expression but ignore case
- -links n - Search for a file with the specified number of links
- -lname name - Search for a symbolic link
- -mmin n - File's data was last modified n minutes ago
- -mtime n - File's data was last modified n days ago
- -name name - Search for a file with the specified name
- -newer name - Search for a file edited more recently than the file given
- -nogroup - Search for a file with no group id
- -nouser - Search for a file with no user attached to it
- -path path - Search for a path
- -readable - Find files which are readable
- -regex pattern - Search for files matching a regular expression
- -type type - Search for a particular type
- -uid uid - Files numeric user id is the same as uid
- -user name - File is owned by user specified
- -writable - Search for files that can be written to

- If you want to find all the empty files and folders in your system use the following command:

  #find / -empty

- If you want to find all of the executable files on your computer use the following command:

  #find / -exec

- To find all of the files that are readable use the following command:

  #find / -read

- How To Send Output From Find The Find Command To A File

The main problem with the find command is that it can sometimes return too many results to look at in one go.You can pipe the output to the tail command or you can output the lines to a file as follows:

  #find / -name *.mp3 -fprint nameoffiletoprintto

- How To Find And Execute A Command Against A File

  Imagine you want to search for and edit a file at the same time.

  You can use the following command:

  #find / -name filename -exec nano '{}' \;

  The above command searches for a file called filename and then runs the nano editor for the file that it finds.

FIND WITH PERMISSION

- Find all the files whose permissions are 777.

  # find . -type f -perm 0777 -print

- Find all the files without permission 777.

  # find / -type f ! -perm 777

- Find all the SGID bit files whose permissions set to 644.

  # find / -perm 2644

- Find all the Sticky Bit set files whose permission are 551.

  # find / -perm 1551

- Find all SUID set files.

```
# find / -perm /u=s
```

- Find all SGID set files.

```
# find / -perm /g=s
```

- Find all Read Only files.

```
# find / -perm /u=r
```

- Find all Executable files.

```
# find / -perm /a=x
```

- Find all 777 permission files and use chmod command to set permissions to 644.

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

- Find all 777 permission directories and use chmod command to set permissions to 755.

```
# find / -type d -perm 777 -print -exec chmod 755 {} \;
```

- To find a single file called tecmint.txt and remove it.

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

- To find and remove multiple files such as .mp3 or .txt, then use.

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

OR

```
# find . -type f -name "*.mp3" -exec rm -f {} \;
```

- To find all empty files under certain path.

```
# find /tmp -type f -empty
```

- To file all empty directories under certain path.

```
# find /tmp -type d -empty
```

- To find all hidden files, use below command.

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

Find Files and Directories Based on Owner

- To find all or single file called tecmint.txt under / root directory of owner root.

```
# find / -user root -name tecmint.txt
```

- To find all files that belongs to user Tecmint under /home directory.

```
# find /home -user tecmint
```

- To find all files that belongs to group Developer under /home directory.

```
# find /home -group developer
```

- To find all .txt files of user Tecmint under /home directory.

```
# find /home -user tecmint -iname "*.txt"
```

Find Files and Directories Based on Date and Time

- To find all the files which are modified 50 days back.

```
# find / -mtime 50
```

- To find all the files which are accessed 50 days back.

```
# find / -atime 50
```

- To find all the files which are modified more than 50 days back and less than 100 days.

```
# find / -mtime +50 –mtime -100
```

- To find all the files which are changed in last 1 hour.

```
# find / -cmin -60
```

- To find all the files which are modified in last 1 hour.

```
# find / -mmin -60
```

- To find all the files which are accessed in last 1 hour.

```
# find / -amin -60
```

Find Files and Directories Based on Size

- To find all 50MB files, use.

```
# find / -size 50M
```

- To find all the files which are greater than 50MB and less than 100MB.

```
# find / -size +50M -size -100M
```

- To find all 100MB files and delete them using one single command.

```
# find / -size +100M -exec rm -rf {} \;
```

- Find all .mp3 files with more than 10MB and delete them using one single command.

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

- Limit depth of directory traversal

The find command by default travels down the entire directory tree recursively, which is time and resource consuming. However the depth of directory travesal can be specified. For example we don't want to go more than 2 or 3 levels down in the sub directories. This is done using the maxdepth option.

```
$ find ./test -maxdepth 2 -name "*.php"
./test/subdir/how.php
./test/cool.php

$ find ./test -maxdepth 1 -name *.php
./test/cool.php
```

The second example uses maxdepth of 1, which means it will not go lower than 1 level deep, either only in the current directory.

This is very useful when we want to do a limited search only in the current directory or max 1 level deep sub directories and not the entire directory tree which would take more time.

Just like maxdepth there is an option called mindepth which does what the name suggests, that is, it will go atleast N level deep before searching for the files.

- Combine multiple search criterias

```
$ find ./test -name 'abc*' ! -name '*.php'
./test/abc.txt
./test/abc
```

The above find command looks for files that begin with abc in their names and do not have a php extension. This is an example of how powerful search expressions can be build with the find command.

- When using multiple name criterias, the find command would combine them with AND operator, which means that only those files which satisfy all criterias will be matched. However if we need to perform an OR based matching then the find command has the "o" switch.

```
$ find -name '*.php' -o -name '*.txt'
```

- So lets say you want to search inside 2 separate directories. Again, the command is very simple

```
$ find ./test ./dir2 -type f -name "abc*"
./test/abc.txt
./dir2/abcdefg.txt
```

- Hidden files on linux begin with a period. So its easy to mention that in the name criteria and list all hidden files.

```
$ find ~ -type f -name ".*"
```

- The following command will display the 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
```

Similary when sorted in ascending order, it would show the smallest files first

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

The following command uses the "empty" option of the find command, which finds all files that are empty.

```
# find /tmp -type f -empty
```

- To file all empty directories use the type "d".

```
$ find ~/ -type d -empty
```

- Lets say we found files using find command, and now want to list them out as the ls command would have done. This is very easy.

```
$ find . -exec ls -ld {} \;
drwxrwxr-x 4 enlightened enlightened 4096 Aug 11 19:01 .
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:25 ./abc.txt
drwxrwxr-x 2 enlightened enlightened 4096 Aug 11 16:48 ./abc
drwxrwxr-x 2 enlightened enlightened 4096 Aug 11 16:26 ./subdir
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:26 ./subdir/how.php
-rw-rw-r-- 1 enlightened enlightened 29 Aug 11 19:13 ./abc.php
-rw-rw-r-- 1 enlightened enlightened 0 Aug 11 16:25 ./cool.php
```

- The following command will remove all text files in the tmp directory.

```
$ find /tmp -type f -name "*.txt" -exec rm -f {} \;
```

- Delete files larger than 100MB

```
$ find /home/bob/dir -type f -name *.log -size +10M -exec rm -f {} \;
```