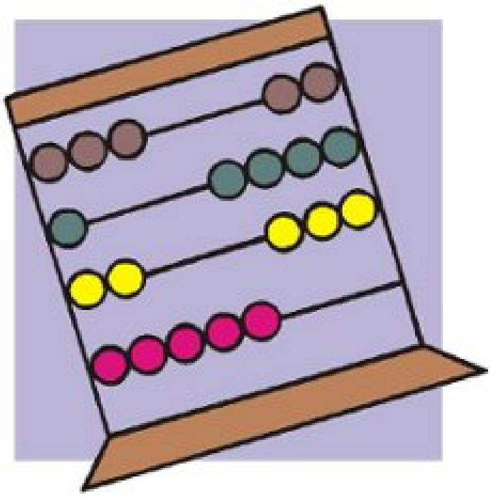
How to count how many files are in a directory with find on Linux

Author : admin



Did you ever needed to count, **how many files in a directory are there?**
Having the concrete number of files in a directory is not a seldom task but still very useful especially for scripts or simply for the sake of learning

The quickest and maybe the easiest way to **count all files in a directory** in Linux is with a combination of **find** and **wc** commands:

Here is how;

```
linux:~# cd ascii
linux:~/ascii# find . -type f -iname '*' -print |wc -l
407
```

This will find and list all matched files in any directory and subdirectories, print them out and count them with **wc** command.
The *-type f* argument instructs find to look only for files.

Other helpful variance of finding and listing all files in a directory and subdirectories is **to list and count all the files with a certain file extension under a directory.** For example, lets list all text files *(.txt)* contained in a directory and all level sub-directories:

```
linux:~/ascii# find . -type f -iname '*.txt' -print |wc -l
401
```

If you need to check the number of files in a directory for multiple directories on a server and you're aiming at doing it efficienly, issung above *find* .. | *wc* code will definitely be not a good choice. If used it will generate heavy load for the system and along with that will complete the execution in ages if issued on a large number of files containing dirs.

Thanksfully if efficiency is targetted, there is a command written in C called **tree** which is more efficient than *find*.
To count the number of files in dir but using *tree* :

```
linux:~# cd ascii
linux:/ascii# tree | tail -n 1
32 directories, 407 files
```

By default *tree* prints info for both the number of found files and directories.
To print out only the files matched, **awk** comes handy, e.g.:

```
linux:/ascii# tree |tail -n 1| awk '{ print $3 }'407
```

To list only the number of files in a directory without its existing sub-directories **ls** + **wc** use is also possible:

```
linux:~/ascii# ls -l | grep ^- | wc -l68
```

This result the above command would produce is +1 more than the real number of files, as it counts the directory ".." as one file (in UNIX / LINUX everything is file).

A short one liner script that can calculate all files correctly by substracting 1 is and hence present correct result on number of files is like so:

```
linux:~/ascii# var=$(ls -l | grep ^- | wc -l); var=$(($var - 1)); echo $var
```

ls can be used to **calculate the number of 1-st level sub-directories under certain directory** for instance:

```
linux:~/ascii# ls -l |grep ^d|wc -l
25
```

You see the **ascii** directory has 25 subdirectories in its 1st level.

To check symlinks under a directory with *ls* the command would be:

```
linux:~/ascii# ls -l | grep ^l | wc -l
```

```
0
```

Note above 3 *ls* | *grep* ... examples, will not work properly if the directory contains files with SUID or some special properties set.
Hence to get the same 3 results for active files, directories and symbolic links, a one liner similar to the one below can be used instead:

```
linux:~/ascii# for t in files links directories; do echo `find . -type ${t:0:1} | wc -l` $t; done 2> /dev/null
407 files
0 links
33 directories
```

This will *show statistics about all files, links and directories for all directory sub-levels.*
Just in case if there is need to only **count files, links and directories** without directory recursion enabled, use:

```
linux:~/ascii# for t in files links directories; do echo `find . -maxdepth 1 -type ${t:0:1} | wc -l` $t; done 2> /dev/null
68 files
0 links
26 directories
```

Anyways the above bash loop will be slow, for directories containing thousands of files. For better performance the equivallent of above bash loop rewritten in perl would be:

```
linux:~/ascii# ls -l |perl -e 'while(){$h{substr($_,0,1)}+=1;} END {foreach(keys %h) {print "$_ $h{$_}\n";}}'
- 68
d 25
t 1
linux:~/ascii#
```

In any case the most preferrable and efficient way to count files en directories is by using **tree** command.
In my view using always tree command instead of code "hacks" is smart idea.

In **Slackware** *tree* command is part of the base install, on **Debian** and **CentOS Linux**, **tree** cmd is not part of the *base system* and requires install via apt / yum e.g.:

```
debian:~# apt-get --yes install tree
...
```

```
[root@centos:~ ]# yum --yes install tree
```

Happy counting ;)