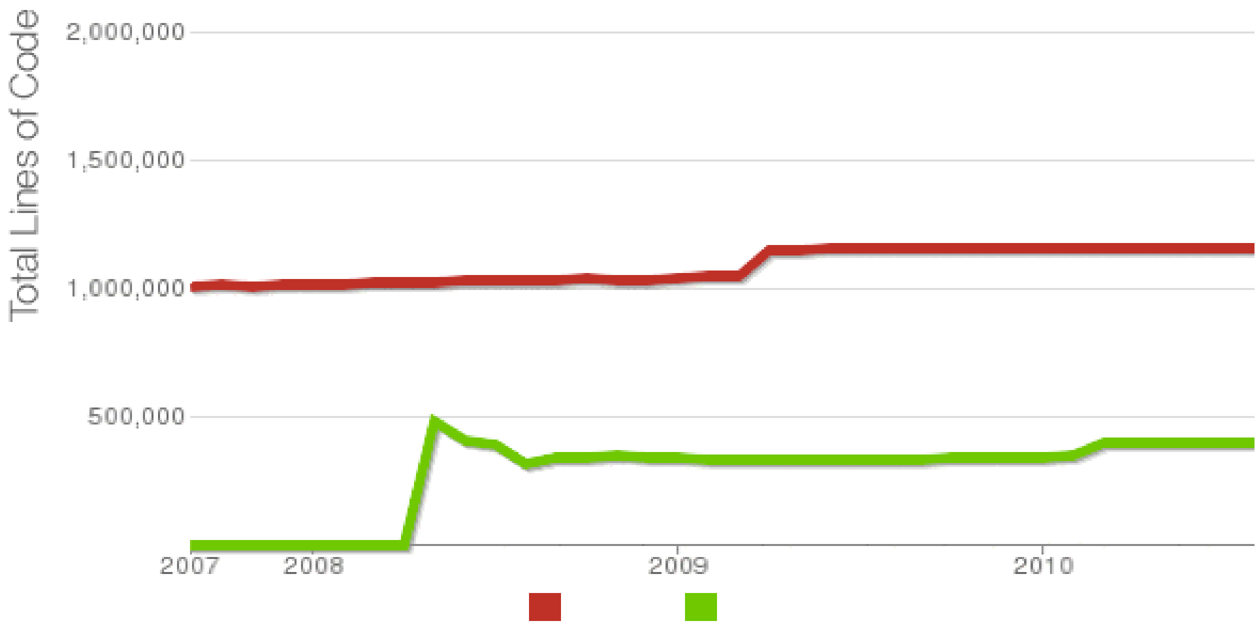
How to count lines of PHP source code in a directory (recursively)

Author : admin



Being able to **count the number of PHP source code lines for a website** is a major statistical information for timely auditting of projects and *evaluating real Project Managment costs*. It is inevitable process for any software project evaluation to count the number of source lines programmers has written. In many small and middle sized software and website development companies, it is the system administrator task to provide information or script quickly something to give info on the exact total number of source lines for projects.

Even for personal use out of curiousity it is useful to know how many lines of PHP source code a wordpress or Joomla website (with the plugins) contains.
Anyone willing to **count the number of PHP source code lines** under one directory level, could do it with:::

```
serbver:~# cd /var/www/wordpress-website
server:/var/www/wordpress-website:# wc -l *.php
17 index.php
101 wp-activate.php
1612 wp-app.php
12 wp-atom.php
19 wp-blog-header.php
105 wp-comments-post.php
12 wp-commentsrss2.php
90 wp-config-sample.php
85 wp-config.php
```

```
104 wp-cron.php
12 wp-feed.php
58 wp-links-opml.php
59 wp-load.php
694 wp-login.php
236 wp-mail.php
17 wp-pass.php
12 wp-rdf.php
15 wp-register.php
12 wp-rss.php
12 wp-rss2.php
326 wp-settings.php
451 wp-signup.php
110 wp-trackback.php
109 xmlrpc.php
4280 total
```

This will count and show statistics, for each and every PHP source file within **wordpress-website** (non-recursively), to get only information about the **total number of PHP source code lines within the directory**, one could **grep it**, e.g.:::

```
server:/var/www/wordpress-website:# wc -l *.php |grep -i '\stotal$'
4280 total
```

The command *grep -i '\stotal$'* has **\s** in beginning and **$** at the end of *total* keyword in order to omit erroneously matching PHP source code file names which contain **total** in file name; for example *total.php* .... *total_blabla.php* .... *blabla_total_bla.php* etc. etc.

The **\s** grep regular expression meaning is *"put empty space"*, "$" is placed at the end of tital to indicate to regexp grep only for words ending in string *total*.

So far, so good ... Now it is most common that instead of counting the PHP source code lines for a first directory level to **count complete number of PHP, C, Python whatever source code lines recursively** - i. e. (a source code of website or projects kept in multiple sub-directories). To **count recursively lines of programming code for any existing filesystem directory** use **find** in conjunction with **xargs**:::

```
server:/var/www/wp-website1# find . -name '*.php' | xargs wc -l
1079 ./wp-admin/includes/file.php
2105 ./wp-admin/includes/media.php
103 ./wp-admin/includes/list-table.php
1054 ./wp-admin/includes/class-wp-posts-list-table.php
105 ./wp-admin/index.php
109 ./wp-admin/network/user-new.php
100 ./wp-admin/link-manager.php
```

```
410 ./wp-admin/widgets.php
108 ./wp-content/plugins/akismet/widget.php
104 ./wp-content/plugins/google-analytics-for-wordpress/wp-gdata/wp-gdata.php
104 ./wp-content/plugins/cyr2lat-slugs/cyr2lat-slugs.php
,,,,
652239 total
```

As you see the cmd counts and displays the number of source code lines encountered in each and every file, for big directory structures the screen gets floated and passing | *less* is nice, e.g.:

```
find . -name '*.php' | xargs wc -l | less
```

Displaying lines of code for each file within the directories is sometimes unnecessery, whether just a **total number of programming source code line** is required, hence for scripting purposes it is useful to only *get the source lines total num*:::

```
server:/var/www/wp-website1# find . -name '*.php' | xargs wc -l | grep -i '\stotal$'
```

Another *shorter and less CPU intensive one-liner* to calculate the lines of codes is:::

```
server:/var/www/wp-website1# ( find ./ -name '*.php' -print0 | xargs -0 cat ) | wc -l
```

Here is one other shell script which displays all file names within a directory with the respective calculated lines of code

For more professional and bigger projects using pure Linux bash and command line scripting might not be the best approach. For counting huge number of programming source code and displaying various statistics concerning it, there are two other tools - SLOCCount
as well as clock (count lines of code)

Both tools, are written in Perl, so for IT managers concerned for speed of calculating projects source (if too frequent source audit is necessery) this tools might be a bit sluggish. However for most projects they should be of a great add on value, actually SLOCCount was already used for calculating the development costs of GNU / Linux and other projects of high importance for Free Software community and therefore it is proven it works well with *ENORMOUS software source line code calculations written in programming languages of heterogenous origin.*

sloccount and *cloc* packages are available in default Debian and Ubuntu Linux repositories, so if you're a Debilian user like me you're in luck:::

```
server:~# apt-cache search cloc$
cloc - statistics utility to count lines of code
```

```
server:~# apt-cache search sloccount$
sloccount - programs for counting physical source lines of code (SLOC)
```

Well that's all folks, Cheers en happy counting ;)