How to replace a string in multiple files in linux command line

Asked 9 years, 11 months ago Modified 8 months ago Viewed 788k times

689

I need to replace a string in a lot of files in a folder, with only `ssh` access to the server. How can I do this?

linux string

201

Share Follow

edited Jun 14, 2017 at 14:38
Drise
4,164 5 38 64

asked Jul 9, 2012 at 9:37
mridul4c
7,075 3 17 27

1 If you'd prefer to use Notepad++ instead of command line, I found this really helpful: superuser.com/a/1003801/74576 – Ryan Sep 8, 2020 at 19:54

27 Answers

Sorted by: Highest score (default)

¿No encuentras la respuesta? Pregunta en Stack Overflow en español.

861

```
cd /path/to/your/folder
sed -i 's/foo/bar/g' *
```

Occurrences of "foo" will be replaced with "bar".

On BSD systems like macOS, you need to provide a backup extension like `-i '.bak'` or else "risk corruption or partial content" per the manpage.

```
cd /path/to/your/folder
sed -i '.bak' 's/foo/bar/g' *
```

Share Follow

edited May 18, 2020 at 19:16
Frungi
507 5 16

answered Jul 9, 2012 at 9:39
kev
147k 41 264 265

14 This doesn't seem to work for me if the string has whitespaces or special characters in it. Any idea why that might be, or do I need to escape them some how? Thanks! – Matthew Herbst Aug 6, 2014 at 17:54

14 At least for my version of `sed`, `-i` requires a string after it, which is appended to the old file names. So `sed -i .bak 's/foo/bar/g' *.xx` moves all `.xx` files to the equivalent `.xx.bak` name and then generates the `.xx` files with the foo→bar substitution. – Anaphory Oct 4, 2014 at 22:35

35 If anybody want to look for more options, there is an answer on unix stack exchange which covers more use cases site unix.stackexchange.com/a/112024/13488 – Reddy May 22, 2015 at 9:40

32 @MatthewHerbst to escape spaces, use \ like `sed -i 's/foo\ with\ spaces/bar/g' *`. I suppose you've found out after so long... but this way it stays for others finding the same issue. – manuelvigarcia Dec 21, 2016 at 8:44

10 For something recursive you could try the following. Note that it doesn't work if the list of files is huge. `sed -i -e 's/foo/bar/g' $(find /home/user/base/dir)` – Scot Mar 28, 2018 at 3:06

306

Similar to Kaspar's answer but with the g flag to replace all the occurrences on a line.

```
find ./ -type f -exec sed -i 's/string1/string2/g' {} \;
```

For global case insensitive:

```
find ./ -type f -exec sed -i 's/string1/string2/gI' {} \;
```

Share Follow

edited Oct 18, 2016 at 14:54
Kellen Stuart
6,353 5 50 73

answered Jan 24, 2014 at 14:45
Céline Aussourd
9,326 4 32 36

27 If you are on OSX and your patterns might contain dots and you want in-place replacement (no backup file) you should use `LC_ALL=C find ./ -type f -exec sed -i '' -e 's/proc.priv/priv/g' {} \;` (see this post and this one) – Jonathan H Aug 13, 2015 at 15:01

2 This definitely worked for me, as it enables filtering with -name "*.js"' – Marcello de Sales Feb 17, 2016 at 11:30

1 A verbose option would be cool, but you can just re-grep to see if the changes were made. Note: For wildcards, try '-name "*.php"' and grep is bad with recursion and wildcards, you need to add `--include=*.whatever` with -r – PJ Brunet Feb 23, 2017 at 18:41

22 Don't do this from the root of a checked-out Git repo. You might accidentally corrupt its database in `.git/`. Be sure to `cd` down to a lower level. – erikprice Mar 7, 2017 at 20:27

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email | Sign up with Google | Sign up with GitHub | Sign up with Facebook

286

@kev's answer is good, but only affects files in the immediate directory.The example below uses grep to recursively find files. It works for me everytime.

```
grep -rli 'old-word' * | xargs -i@ sed -i 's/old-word/new-word/g' @
```

Command breakdown

grep -r: *--recursive*, recursively read all files under each directory.
grep -l: *--print-with-matches*, prints the name of each file that has a match, instead of printing matching lines.
grep -i: *--ignore-case*.

xargs: transform the STDIN to arguments, follow this answer.
xargs -i@ ~command contains @~: a placeholder for the argument to be used in a specific position in the **~command~**, the @ sign is a placeholder which could replaced by any string.

sed -i: edit files in place, **without** backups.
sed s/regexp/replacement/: substitute string matching **regexp** with **replacement**.
sed s/regexp/replacement/**g**: *global*, make the substitution for each match instead of only the first match.

Share Follow

edited Nov 7, 2019 at 6:03
weshouman
495 8 14

answered Dec 21, 2013 at 16:52
pymarco
7,117 4 26 40

17 this did not work for me, but this did: `grep --include={*.php,*.html,*.js} -rnl './' -e "old-word" | xargs -i@ sed -i 's/old-word/new-word/g' @` – Dennis Mar 5, 2014 at 22:20

7 @pymarco Can you please explain this command? I don't know why you had to use xargs instead of just using sed after `|`, also, why `-i` in xargs command? I read in the manual it is deprecated and should use `-I` instead. And is `@` used as a delimiter for beginning and end for pattern? – Edson Horacio Junior Nov 9, 2015 at 14:57

2 the question is specifically for linux though. – pymarco Oct 25, 2016 at 5:22

16 On osx this is ok : `grep -rli 'old-word' * | xargs -I@ sed -i '' 's/2.0.0/latest/g' @` – Philippe Sultan Oct 27, 2017 at 10:14

1 It would be nice if you broke down some of the options (`rli`) and the `@` sign for example – Foobar Aug 27, 2018 at 13:43

55

There are a few standard answers to this already listed. Generally, you can use **find** to recursively list the files and then do the operations with **sed** or **perl**.

rpl

For most quick uses, you may find the command **rpl** is much easier to remember.

Replace `foo` with `bar` on all `.txt` files:

```
rpl -v foo bar '*.txt'
```

Simulate replacing the regex `foo.*` with `bar` in all `.txt` files recursively:

```
rpl --dry-run 'foo.*' bar '**/*.txt'
```

You'll probably need to install it (`apt-get install rpl` or similar).

repren

However, for tougher jobs that involve regular expressions and back substitution, or file renames as well as search-and-replace, the most general and powerful tool I'm aware of is **repren**, a small Python script I wrote a while back for some thornier renaming and refactoring tasks. The reasons you might prefer it are:

- Support renaming of files as well as search-and-replace on file contents.
- See changes before you commit to performing the search and replace.
- Support regular expressions with back substitution, whole words, case insensitive, and case preserving (replace foo -> bar, Foo -> Bar, FOO -> BAR) modes.
- Works with multiple replacements, including swaps (foo -> bar and bar -> foo) or sets of non-unique replacements (foo -> bar, f -> x).

To use it, `pip install repren`. Check the README for examples.

Share Follow

edited Sep 15, 2021 at 15:56
Flimm
116k 38 227 242

answered Mar 22, 2015 at 6:09
jlevy
2,718 1 15 10

3 Wow, repren is excellent! Just used it to change part of a word inside of class names, methods and variables while renaming files to match across 1,000+ C++ header and source files and it worked

49

This worked for me:

```
find ./ -type f -exec sed -i 's/string1/string2/' {} \;
```

Howerver, this did not: `sed -i 's/string1/string2/g' *`. Maybe "foo" was not meant to be string1 and "bar" not string2.

Share Follow

edited Apr 1, 2015 at 13:34
fedorqui
253k 96 511 570

answered Oct 29, 2012 at 13:15
Kaspar L. Palgi
994 8 18

1 It is because sed treats the wildcard * differently. [abc]* means an arbitrary number of characters of the set {a, b, c}. [a-z0-9]* works similar to the wildcard *. – thepiercingarrow Mar 17, 2016 at 0:19

13 On OSX use: `find ./ -type f -exec sed -i '' -e 's/string1/string2/' {} \;` – Shaheen Ghiassy Apr 16, 2018 at 11:03

38

To replace a string in multiple files you can use:

```
grep -rl string1 somedir/ | xargs sed -i 's/string1/string2/g'
```

E.g.

```
grep -rl 'windows' ./ | xargs sed -i 's/windows/linux/g'
```

Source blog

Share Follow

edited Sep 7, 2018 at 13:53

answered May 22, 2014 at 8:09
Djacomo
495 4 9

31

To replace a path within files (avoiding escape characters) you may use the following command:

```
sed -i 's@old_path@new_path@g'
```

The @ sign means that all of the special characters should be ignored in a following string.

6 Exactly what I was looking for in all the other answers. Namely, how to deal with special characters, such as when changing a string that is a path. Thank you. Seemed like a big oversight in the other answers. – inspirednz Feb 28, 2018 at 5:54

30

Given you want to search for the string `search` and replace it with `replace` across multiple files, this is my battle-tested, one-line formula:

```
grep -RiIl 'search' | xargs sed -i 's/search/replace/g'
```

Quick grep explanation:

- `-R` - recursive search
- `-i` - case-insensitive
- `-I` - skip binary files (you want text, right?)
- `-l` - print a simple list as output. Needed for the other commands

The grep output is then piped to sed (through xargs) which is used to actually replace text. The `-i` flag will alter the file directly. Remove it for a kind of "dry run" mode.

Share Follow

answered Aug 26, 2019 at 10:22
Ignorant
2,160 4 26 39

3 If the text happens to be a url, remember you use different delimiters with sed ike this: 's#search#replace#g' – Jérôme Tremblay Jan 14 at 21:42

1 When using `grep -i` for case-insensitive search, you also need to let sed do its own search in a case-insensitive way. For that, replace the sed command with `sed -i 's/search/replace/gI'` . – tanius Mar 21 at 2:48

22

In case your string has a forward slash(/) in it, you could change the delimiter to '+'.

```
find . -type f -exec sed -i 's+http://example.com+https://example.com+g' {} +
```

This command would run recursively in the current directory.

Share Follow

answered Dec 14, 2017 at 14:04
gopiariv

13

If you have list of files you can use

```
replace "old_string" "new_string" -- file_name1 file_name2 file_name3
```

If you have all files you can use

```
replace "old_string" "new_string" -- *
```

If you have list of files with extension, you can use

```
replace "old_string" "new_string" -- *.extension
```

Share Follow

edited Jan 8, 2018 at 9:39
Jawahar Rakkiannan
3 2

answered May 8, 2014 at 19:10
Pawel Dubiel
16.3k 3 39 55

2 actually, "--file" should just be "--", at least in my version – simpleuser Oct 7, 2014 at 23:34

4 This utility is distributed within MySQL packages. – Dmitry Ginzburg Oct 14, 2014 at 16:00

2 Though I would like to appreciate solution, which works without quoting regular line to be regular expression. – Dmitry Ginzburg Oct 14, 2014 at 16:05

1 This works fine - and if you want to replace all strings in multiple files, which end in e.g. ".txt" , you can just do `replace "old_string" "new_string" -- *.txt` – tsveti_iko Apr 6, 2016 at 12:56

2 It's better to add from where to get this `replace` utility. Without this information, this answer is incomplete. – Sitesh Apr 3, 2018 at 4:34

12

The first line occurrences of "foo" will be replaced with "bar". And you can using the second line to check.

```
grep -rl 'foo' . | xargs sed -i 's/foo/bar/g'
grep 'foo' -r * | awk -F: {'print $1'} | sort -n | uniq -c
```

Share Follow

edited Dec 18, 2017 at 11:05
Gogol BH Network
2,944 4 27 52

answered Mar 20, 2017 at 8:30
jiasir
121 1 4

6 Why are you linking to to your blog? It contains exactly the same text as your answer. – DavidPostill

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email | Sign up with Google | Sign up with GitHub | Sign up with Facebook

10

"You could also use find and sed, but I find that this little line of perl works nicely.

```
perl -pi -w -e 's/search/replace/g;' *.php
```

- -e means execute the following line of code.
- -i means edit in-place
- -w write warnings
- -p loop

" (Extracted from http://www.liamdelahunty.com/tips/linux_search_and_replace_multiple_files.php)

My best results come from using perl and grep (to ensure that file have the search expression)

```
perl -pi -w -e 's/search/replace/g;' $( grep -rl 'search' )
```

Share Follow

answered Jan 16, 2015 at 13:56
Alejandro Salamanca Mazuelo
1,063 15 18

I prefer this as its cross-platform with no effort between Mac/Linux – Ben Apr 23 at 22:41

6

On a MacBook Pro, I used the following (inspired by https://stackoverflow.com/a/19457213/6169225):

```
sed -i '' -e 's/<STR_TO_REPLACE>/<REPLACEMENT_STR>/g' *
```

`-i ''` will ensure you are taking no backups.

`-e` for modern regex.

Share Follow

answered Jan 28, 2019 at 17:21
Marquistador
1,631 18 23

5

I did concoct my own solution before I found this question (and answers). I searched for different combinations of "replace" "several" and "xml," because that was my application, but did not find this particular one.

My problem: I had spring xml files with data for test cases, containing complex objects. A refactor on the java source code changed a lot of classes and did not apply to the xml data files. In order to save the test cases data, I needed to change all the class names in all the xml files, distributed across several directories. All while saving backup copies of the original xml files (although this was not a must, since version control would save me here).

I was looking for some combination of `find` + `sed`, because it worked for me in other situations, but not with several replacements at once.

Then I found ask ubuntu response and it helped me build my command line:

```
find -name "*.xml" -exec sed -s --in-place=.bak -e 's/firstWord/newFirstWord
/g;s/secondWord/newSecondWord/g;s/thirdWord/newThirdWord/g' {} \;
```

And it worked perfectly (well, my case had six different replacements). But please note that it will touch all *.xml files under current directory. Because of that, and if you are accountable to a version control system, you might want to filter first and only pass on to `sed` those actually having the strings you want; like:

```
find -name "*.xml" -exec grep -e "firstWord" -e "secondWord" -e "thirdWord" {}
\; -exec sed -s --in-place=.bak -e 's/firstWord/newFirstWord/g;s/secondWord
/newSecondWord/g;s/thirdWord/newThirdWord/g' {} \;
```

Share Follow

edited Apr 13, 2017 at 12:22
Community Bot
1 1

answered Mar 29, 2016 at 7:03
manuelvigarcia
1,166 1 14 27

and for windows, I just found out that there is a way to find the string --didn't check yet how to replace it-- in one command: `findstr /spin /c:"quéquieresbuscar" *.xml` It will be handy.
– manuelvigarcia Jun 8, 2016 at 11:07

5

Really lame, but I couldn't get any of the sed commands to work right on OSX, so I did this dumb thing instead:

```
:%s/foo/bar/g
:wn
```

^- copy these three lines into my clipboard (yes, include the ending newline), then:

vi *

and hold down command-v until it says there's no files left.

Dumb...hacky...effective...

Share Follow

answered Aug 24, 2017 at 22:02

Justin Killen
720 6 19

4

```
grep --include={*.php,*.html} -rnl './' -e "old" | xargs -i@ sed -i 's/old
/new/g' @
```

Share Follow

answered Mar 5, 2014 at 22:20

Dennis
7,455 10 56 109

3

script for multiedit command

```
multiedit [-n PATTERN] OLDSTRING NEWSTRING
```

From Kaspar's answer I made a bash script to accept command line arguments and optionally limit the filenames matching a pattern. Save in your $PATH and make executable, then just use the command above.

Here's the script:

```
#!/bin/bash
_help="\n
Replace OLDSTRING with NEWSTRING recursively starting from current directory\n
multiedit [-n PATTERN] OLDSTRING NEWSTRING\n

[-n PATTERN] option limits to filenames matching PATTERN\n
Note: backslash escape special characters\n
Note: enclose STRINGS with spaces in double quotes\n
Example to limit the edit to python files:\n
multiedit -n \*.py \"OLD STRING\" NEWSTRING\n"

# ensure correct number of arguments, otherwise display help...
if [ $# -lt 2 ] || [ $# -gt 4 ]; then echo -e $_help ; exit ; fi
if [ $1 == "-n" ]; then  # if -n option is given:
        # replace OLDSTRING with NEWSTRING recursively in files matching PATTERN
        find ./ -type f -name "$2" -exec sed -i "s/$3/$4/g" {} \;
else
        # replace OLDSTRING with NEWSTRING recursively in all files
        find ./ -type f -exec sed -i "s/$1/$2/" {} \;
fi
```

Share Follow

edited Nov 8, 2014 at 14:54

answered Nov 8, 2014 at 1:51

BBW Before Windows
578 5 6

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email | Sign up with Google | Sign up with GitHub | Sign up with Facebook

2

The stream editor does modify multiple files “inplace” when invoked with the `-i` switch, which takes a backup file ending as argument. So

```
sed -i.bak 's/foo/bar/g' *
```

replaces `foo` with `bar` in all files in this folder, but does not descend into subfolders. This will however generate a new `.bak` file for every file in your directory. To do this recursively for all files in this directory and all its subdirectories, you need a helper, like `find`, to traverse the directory tree.

```
find ./ -print0 | xargs -0 sed -i.bak 's/foo/bar/g' *
```

`find` allows you further restrictions on what files to modify, by specifying further arguments like `find ./ -name '*.php' -or -name '*.html' -print0`, if necessary.

Note: GNU `sed` does not require a file ending, `sed -i 's/foo/bar/g' *` will work, as well; FreeBSD `sed` demands an extension, but allows a space in between, so `sed -i .bak s/foo/bar/g *` works.

Share Follow

answered Oct 4, 2014 at 23:00
Anaphory
5,619 4 34 64

2

To maintain my personal English node, I wrote an utility script that help to replace multiple pair of old/new string, for all files under a directory recursively.

The multiple pair of old / new string are managed in a hash map.

The dir can be set via command line or environment variable, the map is hard coded in the script, but you can modify the code to load from a file, if necessary.

It requires bash 4.2, due to some new feature.

en_standardize.sh:

```
#! /bin/bash
# (need bash 4.2+,)
#
# Standardize phonetic symbol of English.
#
# format:
#   en_standardize.sh [<dir>]
#
# params:
# * dir
#   target dir, optional,
#   if not specified then use environment variable "$node_dir_en",
#   if both not provided, then will not execute,
# *
#

paramCount=$#

# figure target dir,
if [ $paramCount -ge 1 ]; then # dir specified
    echo -e "dir specified (in command):\n\t$1\n"
    targetDir=$1
elif [[ -v node_dir_en ]]; then # environable set,
    echo -e "dir specified (in environment vairable):\n\t$node_dir_en\n"
    targetDir=$node_dir_en
else # environable not set,
    echo "dir not specified, won't execute"
    exit
fi

# check whether dir exists,
if [ -d $targetDir ]; then
    cd $targetDir
else
    echo -e "invalid dir location:\n\t$targetDir\n"
    exit
fi
```

Share Follow

answered Sep 24, 2016 at 20:11

2

I'd just like to add a note to do two things at once - find a file that contains a string and then do a replace, using the find 'chaining' method:

```
find  . -type f -iname \*.php -exec fgrep -l "www." {} \; -exec sed -i
"s|www||g" {} \;
```

- In this real case, remove the anachronistic 'www' from urls found in PHP files.
- The 'fgrep -l' only triggers if it finds at least one match in a file, it produces no other output. Don't forget the '\;' separators!

Share Follow

answered Nov 10, 2020 at 16:32
Mark
86 4

1

If the file contains backslashes (paths usually) you can try something like this:

```
sed -i -- 's,<path1>,<path2>,g' *
```

ex:

```
sed -i -- 's,/foo/bar,/new/foo/bar,g' *.sh (in all shell scripts available)
```

Share Follow

edited Apr 17, 2015 at 17:17

answered Apr 16, 2015 at 23:14
Shaik Amanulla
11 2

1

Using the ack command would be alot faster like this:

```
ack '25 Essex' -l | xargs sed -i 's/The\ fox \jump/abc 321/g'
```

Also if you have a white space in the search result. You need to escape it.

Share Follow

answered Jan 29, 2019 at 19:44
Patoshi パトシ
19.3k 2 24 40

1

There is an easier way by using a simple script file:

```
# sudo chmod +x /bin/replace_string_files_present_dir
```

open the file in gedit or an editor of your choice, I use gedit here.

```
# sudo gedit /bin/replace_string_files_present_dir
```

Then in the editor paste the following in the file

```
#!/bin/bash
replace "oldstring" "newstring" -- *
replace "oldstring1" "newstring2" -- *
#add as many lines of replace as there are your strings to be replaced for
#example here i have two sets of strings to replace which are oldstring and
#oldstring1 so I use two replace lines.
```

Save the file, close **gedit**, then exit your terminal or just close it and then start it to be able load the new script you added.

Navigate to the directory where you have multiple files you want to edit. Then run:

```
#replace_string_files_present_dir
```

Press enter and this will automatically replace the **oldstring** and **oldstring1** in all the files that contain them with the correct **newstring** and **newstring1** respectively.

It will skip all the directories and files that don't contain the old strings.

This might help in eliminating the tedious work of typing in case you have multiple directories with files that need string replacement. All you have to do is navigate to each of those directories, then run:

```
#replace_string_files_present_dir
```

All you have to do is to ensure you've included or added all the replacement strings as I showed you above:

```
replace "oldstring" "newstring" -- *
```

at the end of the file **/bin/replace_string_files_present_dir**.

oldstring is not found.

Share Follow

answered Jul 31, 2019 at 9:16
briancollins081
668 7 17

1 Usually, when people ask "how to ${whatever}" in bash, they are asking for a compact version to include in a script or **CI jobs** instruction (say a `.gitlab-ci.yml` or a `travis.yml`). Every turn around consisting in writing a script to execute it later is an anti-pattern, for you'll then need to script the script creation (and I goes messy, most of the time) – zar3bski Apr 30, 2020 at 13:05

@zar3bski wut? You really should be putting your individual CI steps in separate shell files so that they are portable to other CI systems. It also makes them testable locally. – airtonix Apr 2, 2021 at 7:20

0

I am giving an example for fixing a common shebang error in python sources.

You can try the grep/sed approach. Here is one that works with GNU sed and won't break a git repo:

```
$ grep -rli --exclude '*.git*' '#!/usr/bin/python' . | xargs -I {} \
gsed -i '' -e 's/#!\/usr\/bin\/python/#!\/usr\/bin\/env python/' {}
```

Or you can use greptile :)

```
$ greptile -x .py -l -i -g '#!/usr/bin/env python' -r '#!/usr/bin/python' .
```

I just tested the first script, and the second should work as well. Be careful with escape characters, I think it should be easier to use greptile in most cases. Of course, you can do many interesting things with sed, and for that it may be preferable to master using it with xargs.

Share Follow

answered Feb 23, 2017 at 20:16
exa
41 3

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up with email | Sign up with Google | Sign up with GitHub | Sign up with Facebook

0

I found this one from another post (can't remember which) and while not the most elegant, it's simple and as a novice Linux user has given me no trouble

```
for i in *old_str* ; do mv -v "$i" "${i/\old_str/new_str}" ; done
```

if you have spaces or other special characters use a \

```
for i in *old_str\ * ; do mv -v "$i" "${i/\old_str\ /new_str}" ; done
```

for strings in sub-directories use **

```
for i in *\*old_str\ * ; do mv -v "$i" "${i/\old_str\ /new_str}" ; done
```

Share Follow

answered Mar 29, 2017 at 16:01
Kyle Turck
43 6

0

Below command can be used to first search the files and replace the files:

```
find . | xargs grep 'search string' | sed 's/search string/new string/g'
```

For example

```
find . | xargs grep abc | sed 's/abc/xyz/g'
```

Share Follow

edited May 17, 2019 at 13:45
Benjamin W.
38.7k 16 96 104

answered Jun 5, 2015 at 7:12
Amit
11 1

0

My problem with many of the answers was that I needed to replace a filepath inside of many files. Though one answer provided mentioned this, it did not work for me. My solution:

First, generate a list of the filenames that you want to be changed.

```
filelist=($(find /path/to/your/folder | xargs grep '/path/to/fix' | cut -d : -f
1 | tr '\n' ' '))
```

What the commands above do is that the `find` piped to `grep` generates the names of the files with the `/path/to/fix` inside. However, `grep` also prints out the line that the string was found on, so the `cut` command gets rid of this and just keeps the filename. `tr` replaces newline characters with spaces, which allows for `filelist` to be stored as an array.

```
for file in "${filelist[@]}"; do sed -i.bak 's+/path/to/fix+/new/path/for/my
/file+g' $file; done
```

This `sed` command draws on other answers to this question, and uses `+` as delimiters rather than the normal `/`, since the `/` characters are being used in the filepath.

Share Follow

answered Jul 6, 2021 at 0:50

Phillip Long
41 7

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.