How to make Windows command prompt treat single quote as though it is a double quote?

Asked 11 years, 5 months ago Modified 8 years, 3 months ago Viewed 22k times

10

My scenario is simple - I am copying script samples from the Mercurial online book and pasting them in a Windows command prompt. The problem is that the samples in the book use single quoted strings. When a single quoted string is passed on the Windows command prompt, the latter does not recognize that everything between the single quotes belongs to one string.

For example, the following command:

```
hg commit -m 'Initial commit'
```

cannot be pasted as is in a command prompt, because the latter treats `'Initial commit'` as two strings - `'Initial` and `commit'`. I have to edit the command after paste and it is annoying.

Is it possible to instruct the Windows command prompt to treat single quotes similarly to the double one?

EDIT

Following the reply by JdeBP I have done a little research. Here is the summary:

- Mercurial entry point looks like so (it is a python program):

  ```
  def run():
      "run the command in sys.argv"
      sys.exit(dispatch(request(sys.argv[1:])))
  ```

- So, I have created a tiny python program to mimic the command line processing used by mercurial:

  ```
  import sys
  print sys.argv[1:]
  ```

- Here is the Unix console log:

  ```
  [hg@Quake ~]$ python 1.py  "1 2 3"
  ['1 2 3']
  [hg@Quake ~]$ python 1.py  '1 2 3'
  ['1 2 3']
  [hg@Quake ~]$ python 1.py  1 2 3
  ```

```
['1', '2', '3']
[hg@Quake ~]$
```

- And here is the respective Windows console log:

  ```
  C:\Work>python 1.py  "1 2 3"
  ['1 2 3']

  C:\Work>python 1.py  '1 2 3'
  ["'1", '2', "3'"]

  C:\Work>python 1.py  1 2 3
  ['1', '2', '3']

  C:\Work>
  ```

One can clearly see that Windows does not treat single quotes as double quotes. And this is the essence of my question.

windows command-line

Share Improve this question

Follow

edited Aug 25, 2013 at 5:30
doubleDown
1,002 ● 8 ● 10

asked Aug 17, 2011 at 11:05
mark
826 ● 4 ● 13 ● 26

2 Either you tagged this question as "Windows" in error, or these are very much *not* DOS command prompts. – JdeBP Aug 17, 2011 at 13:25

Note-Your demo program adds single quotes in its output,which perhaps obscures things a bit.Indeed windows n unix do treat single n double quotes differently though,see my comment in reply2 JdeBP.That said,there r 2 ways Windows programs can get the command line(the area after the prompt).1)GetCommandLine() 2)argsv. argsv probably works by doing GetCommandLine, and the program before you've reached the first line of the main function,will have split it into arguments based on these rules(in the case of Ms Visual C),or else,rules like them. msdn.microsoft.com/en-us/library/a1y7w461.aspx – barlop Aug 25, 2013 at 9:09

try compiling this pastebin.com/raw.php?i=4Bq7unGd and this pastebin.com/raw.php?i=QkhBJWGb Both in Windows. Try calling them with arguments, and see from their output, how they treat arguments differently. One uses GetCommandLine() one uses argsv. Windows itself, doesn't separate out the arguments, doesn't produce the elements of argsv. – barlop Aug 25, 2013 at 9:09

3 Answers

Sorted by: Highest score (default)

10

The quoting character can't be changed in the command.com prompt. You can, however, use PowerShell which accepts both single and double quotes as quoting characters. They function the same as in Unix shells. I.e., single quotes do not expand variables while double quotes will.

You might still run into problems with quotes inside quotes. For example, I have strawberry perl installed on my Windows computer. When I run `perl -e 'print time, "\n" '` in

PowerShell, I see output such as `1321375663SCALAR(0x15731d4)`. I have to escape the double quotes for it to work as expected: `perl -e 'print time, \"\n\" '`

Share Improve this answer Follow

answered Nov 15, 2011 at 16:52
Starfish
928 ● 8 ● 15

4

First, a command prompt is not a command interpreter. (A command prompt is the thing *displayed by* a command interpreter.) Second, your command interpreter, the prompts that it issues, and Win32 consoles, have *nothing at all* to do with this.

In Win32 programs, splitting up the command line into "words" — the NUL-terminated multi-byte character strings that programs in the C and C++ languages see as the argument array passed to `main()` — is the province of the runtime libraries of those programs. On Unices and Linux, the shell does the word splitting, because the operating system actually works in terms of an argument string array. This is not the case for Win32. On Win32, the operating system itself operates in terms of a *command tail*: a *single* long string that still contains all of the quotation marks that one originally typed on the command line. (There *is* some processing done to this command tail by a command interpreter before it is passed to the target program, but it isn't related to word splitting.)

In your case, the runtime library for your `hg` program is being delivered this command tail:

```
commit -m 'Initial commit'
```

The runtime library that that program was compiled with doesn't know that you meant a single quotation mark to be a whitespace quoting character, because *that isn't the convention*. The convention deals *only* in double quotation marks (and backslashes before double quotation marks).

This convention is built into the runtime library that was provided with the compiler used to create the program in the first place. If you want to change the convention, you'll have to re-link *every individual program that you want to run this way* with a special runtime library of your own making that recognizes single quotation marks as well. Clearly this is impractical (unless these are all Cygwin programs).

A far more practical approach is to do what you're already doing: recognize that Windows isn't Unix, and adjust the examples accordingly before using them.

Share Improve this answer Follow

answered Aug 27, 2011 at 12:35
JdeBP
26.3k ● 1 ● 69 ● 103

1 I find it hard to understand the relevance of your answer. Anyway, I have updated my question.
– mark Aug 27, 2011 at 18:33

It's not all down to the runtimes though 'cos even though the program gets it all in one load, the cmd interpreter itself still interprets single quotes differently from double quotes. e.g. if you pass "&" then & is literal and "&" is passed as a parameter, it's different to '&' as the ampersand-& will be treated as special and won't be passed. Also cmd /? has a bunch of rules which I suppose are of the cmd interpreter not runtimes. – barlop Aug 25, 2013 at 8:27

Please consider adding references to back up your claims; and perhaps briefly explain how this picture changes with the PowerShell interpreter. – Olivier Cailloux Mar 11, 2020 at 9:49

+1 for saying *Unices* :D – johny why Sep 17, 2021 at 20:29

0

I'm quite sure that you can't edit the way that DOS parses commands. It's inherent in it's base programming.

The only solution I can think of to speed things up, is keeping a Notepad window open and running a 'Find and Replace' -- replacing all single quotes with with double quotes. And then copy-pasting into DOS from there.

Share Improve this answer Follow

answered Aug 17, 2011 at 11:17

akseli
4,063 ● 19 ● 25

1 I don't know about 9X's command.com(I suppose that is DOS! and 9X is probably still an addin for DOS), but NT's CMD.EXE is not DOS! DOS is an operating system. CMD.EXE is an addin in the NT operating system and won't run outside of it. I have given a -1 for calling it DOS, it is extremely amateurish to call NT's cmd.exe/'windows command processor' "DOS". – barlop Aug 25, 2013 at 8:36