# What's up with the strange treatment of quotation marks and backslashes by CommandLineToArgvW

devblogs.microsoft.com/oldnewthing/20100917-00

September 17, 2010

Raymond Chen

The way the `CommandLineToArgvW` function treats quotation marks and backslashes has raised eyebrows at times. Let's look at the problem space, and then see what algorithm would work. Here are some sample command lines and what you presumably want them to be parsed as:

| Command line | Result |
|---|---|
| `program.exe "hello there.txt"` | `program.exe`<br>`hello there.txt` |
| `program.exe "C:\Hello there.txt"` | `program.exe`<br>`C:\Hello there.txt` |

In the first example, we want quotation marks to protect spaces. In the second example, we want to be able to enclose a path in quotation marks to protect the spaces. Backslashes inside the path have no special meaning; they are copied as any other normal character. So far, the rule is simple: Inside quotation marks, just copy until you see the matching quotation marks. Now here's another wrinkle:

| Command line | Result |
|---|---|
| `program.exe "hello\"there"` | `program.exe`<br>`hello"there` |

In the third example, we want to embed a quotation mark inside a quotated string by protecting it with a backslash. Okay, to handle this case, we say that a backslash which precedes a quotation mark protects the quotation mark. The backslash itself should disappear; its job is to protect the quotation mark and not to be part of the string itself. (If we kept the backslash, then it would not be possible to put a quotation mark into the command line parameter without a preceding backslash.) But what if you wanted a backslash at the end of the string? Then you protect the backslash with a backslash, leaving the quotation mark unprotected.

| Command line | Result |
|---|---|
| `program.exe "hello\\"` | `program.exe hello\` |

Okay, so what did we come up with? We want a backslash before a quotation mark to protect the quotation mark, and we want a backslash before a backslash to protect the backslash (so you can end a string with a backslash). Otherwise, we want the backslash to be given no special treatment. The `CommandLineToArgvW` function therefore works like this:

- A string of backslashes not followed by a quotation mark has no special meaning.
- An even number of backslashes followed by a quotation mark is treated as pairs of protected backslashes, followed by a word terminator.
- An odd number of backslashes followed by a quotation mark is treated as pairs of protected backslashes, followed by a protected quotation mark.

The backslash rule is confusing, but it's necessary to permit the very important second example, where you can just put quotation marks around a path without having to go in and double all the internal path separators. Personally, I would have chosen a different backslash rule:

> **Warning – these are not the actual backslash rules. These are Raymond's hypothetical "If I ran the world" backslash rules.**
>
> - A backslash followed by another backslash produces a backslash.
> - A backslash followed by a quotation mark produces a quotation mark.
> - A backslash followed by anything else is just a backslash followed by that other character.

I prefer these rules because they can be implemented by a state machine. On the other hand, it makes quoting regular expressions a total nightmare. It also breaks `"\\server\share\path with spaces"`, which is pretty much a deal-breaker. Hm, perhaps a better set of rules would be

> **Warning – these are not the actual backslash rules. These are Raymond's second attempt at hypothetical "If I ran the world" backslash rules.**
>
> - Backslashes have no special meaning at all.
> - If you are outside quotation marks, then a `"` takes you inside quotation marks but generates no output.
> - If you are inside quotation marks, then a sequence of 2N quotation marks represents N quotation marks in the output.
> - If you are inside quotation marks, then a sequence of 2N+1 quotation marks represents N quotation marks in the output, and then you exit quotation marks.

This can also be implemented by a state machine, and quoting an existing string is very simple: Stick a quotation mark in front, a quotation mark at the end, and double all the internal quotation marks. But what's done is done, and the first set of backslash rules is what `CommandLineToArgvW` implements. And since the behavior has been shipped and documented, it can't change. If you don't like these parsing rules, then feel free to write your own parser that follows whatever rules you like.

**Bonus chatter**: Quotation marks are even more screwed up.

Raymond Chen

**Follow**