

[Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Topic-wise Practice](#) [C++](#) [Java](#) [Python](#)

Bash Script – Quotes and its types

Last Updated : 30 Jan, 2022



Quoting in bash scripting is a process to make variables or any other container for data to be expanded to the literal value inside any string, quoting is also used for other operations as well. There are several types of quoting techniques in a bash script. In this article, we will see the different types of quoting in Bash scripting.

Types of Quotes in BASH

- Single Quotes
- Double Quotes
- Back Quotes

Single Quotes:

By using single quotes the string is parsed as it is without any expansion of characters inside the quotes. So, if we have a variable inside the string the value won't be expanded instead the characters will be parsed as it is.

Let's say we have a variable called name and we try to echo its value inside a single quote, we need to use `$` to access the variable value. Bu

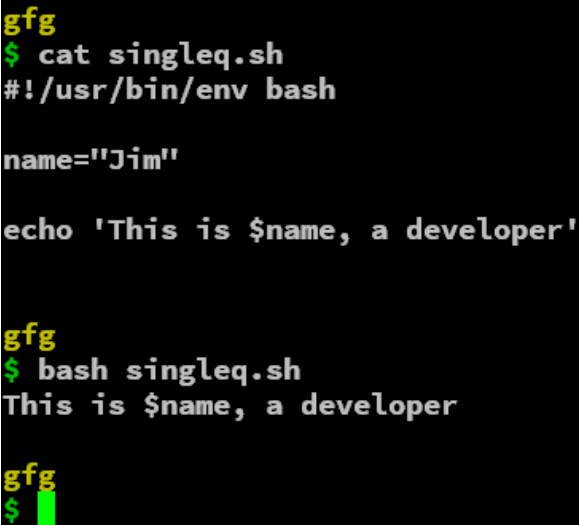
We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

```
name="Jim"
```

```
echo 'This is $name, a developer'
```

Output:

A terminal window with a black background and green text. The prompt 'gfg' is shown in yellow. The user enters '\$ cat singleq.sh' and the shell outputs '#!/usr/bin/env bash' followed by 'name="Jim"' and 'echo \'This is \$name, a developer\''. Then, the user enters '\$ bash singleq.sh' and the shell outputs 'This is \$name, a developer'. Finally, the user enters '\$' and the prompt returns.

```
gfg
$ cat singleq.sh
#!/usr/bin/env bash

name="Jim"

echo 'This is $name, a developer'

gfg
$ bash singleq.sh
This is $name, a developer

gfg
$
```

So, single quotes are not great if you wanted to use the variable expansion and other related stuff. You can close the quotes before the variable and then open again just like concatenation but it's not great for multiple variables.

Double Quotes:

To expand the variable value inside a string we use the double-quotes. Using double quotes we can expand the literal value of the variable by just prefixing the variable name with \$ as said earlier for accessing the value with the variable name.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

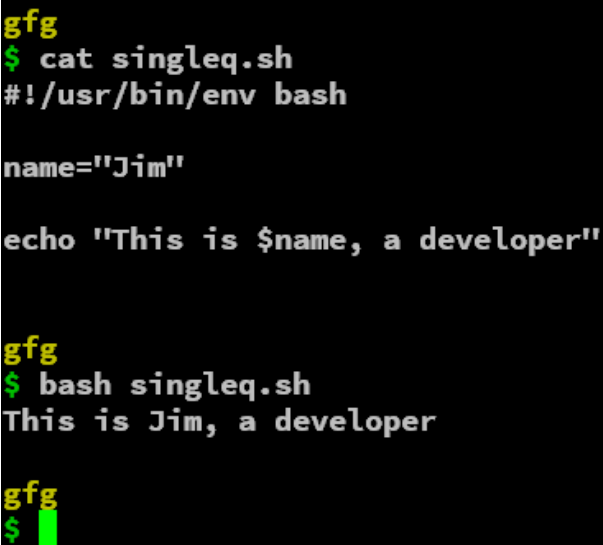
Got It !

```
#!/usr/bin/env bash

name="Jim"

echo "This is $name, a developer"
```

Output:



```
gfg
$ cat singleq.sh
#!/usr/bin/env bash

name="Jim"

echo "This is $name, a developer"

gfg
$ bash singleq.sh
This is Jim, a developer

gfg
$
```

So, as we can see the variable value was expanded and the command worked as expected. The variable value can be anything like integers or strings or any characters that are not expandable further.

Not only variable expansion, the other characters that are special when embedded inside double-quotes.

```
#!/usr/bin/env bash

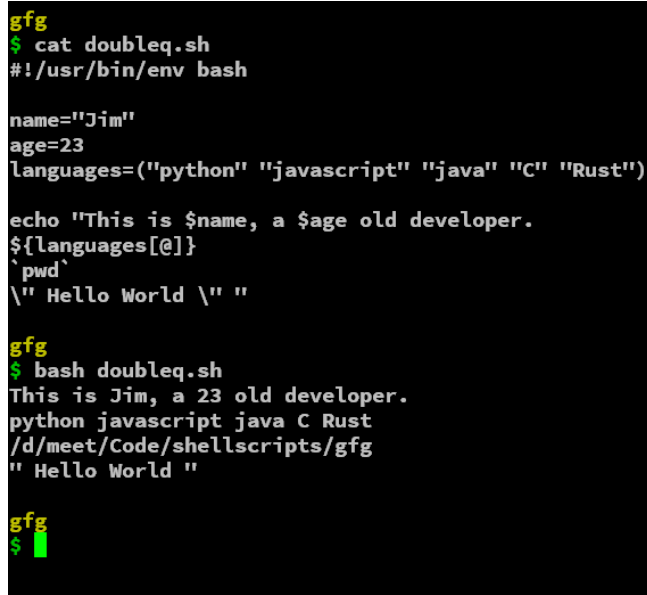
name="Jim"
age=23
languages=("python" "javascript" "java" "C" "Rust")
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

```
`pwd`  
\" Hello W0\orld \" "
```

Output:



```
gfg  
$ cat doubleq.sh  
#!/usr/bin/env bash  
  
name="Jim"  
age=23  
languages=("python" "javascript" "java" "C" "Rust")  
  
echo "This is $name, a $age old developer."  
${languages[@]}  
`pwd`  
\" Hello World \" "  
  
gfg  
$ bash doubleq.sh  
This is Jim, a 23 old developer.  
python javascript java C Rust  
/d/meet/Code/shellscripts/gfg  
" Hello World "  
  
gfg  
$
```

So, this is how we can perform variable expansion in a double quote, we can even escape characters in double quotes by using a backslash. The backslash operator allows parsing the next character as it is without considering it as a special character for expansion or as a wildcard operator.

We have used the variable name and age so as to demonstrate the expansion of different types of data variables inside the double-quotes. We have even used operators to print the entire array namely the `${array_name[@]}`. We have also added backquotes (``) to use a command inside double-quotes. And finally, the backslash character to escape those characters in this case the double quotes ("). Escaping a character is to parse the character in such a way that it is just treated as an ordinary character without allowing its operations as the special

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

Escaping Characters in double quotes

To escape characters we can use `\` before the character to parse it as it is. For example, you need to print the `$` symbol you need type `\$` to work as desired inside the double quotes. We have used `\"` to escape the double quotes and by that, we have used double quotes inside the double quotes. So, to parse the character without its special operations to perform we need to use `\` to simply print it.

Back Quotes(Backticks):

In the previous section, we have used the backquotes to use the commands inside the double-quotes. The back quotes allow us to execute the commands inside a shell script, Not only they can be used inside the double quotes they can be used independently in the script.

```
#!/usr/bin/env bash
```

```
echo `python --version`  
echo `cat wh.txt`  
echo `date`
```

Output:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

```
gfg
$ cat backq.sh
#!/usr/bin/env bash

echo `python --version`
echo `cat wh.txt`
echo `date`

gfg
$ bash backq.sh
Python 3.7.6
Line 1 Some text 98654321 This is a test input
Wed, Jan 26, 2022 3:08:07 PM

gfg
$
```

As we can see here we can execute commands from the script. These are the commands which can be executed inside the command prompt, yes you can use certain commands like cat, pwd, etc as it is in the script but not all commands can be used as it is in the script. For executing those commands we can use the back quotes(` `). These can be also be nested in double-quotes as we saw earlier in the double quotes example.

So these are the quoting techniques we use in the shell script to expand variables, execute commands and perform and execute other operations/scripts from a shell script. The quoting can be used for programming certain commands and executing/expanding them as and when required by embedding the commands in the appropriate quotes.

MASTER CODING WITH 
Daily Problem Of The Day | Weekly Interview Series | Curated Practice Sheets

Start Learning

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

[Previous](#)[Next](#)

Brutal - Create various Payload, PowerShell Attack, Virus Attack and Launch Listener for a HID

How to Create SSH Tunneling or Port Forwarding in Linux?

RECOMMENDED ARTICLES

Page : [1](#) [2](#) [3](#)

01 Bash Script - Difference between Bash Script and Shell Script
23, Jan 22

05 Difference Between Single and Double Quotes in Shell Script and Linux
14, Nov 20

02 Bash Script - Define Bash Variables and its types
16, Mar 22

06 Bash Scripting - Difference between Zsh and Bash
01, Jun 22

03 Bash Scripting - Introduction to Bash and Bash Scripting
19, Apr 22

07 Bash Scripting - Bash Read Password without Echoing back
24, Nov 21

04 Bash Script - Working of Bash Variables
30, Apr 22

08 Bash Scripting - Bash Echo Command
19, Nov 21

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

Article Contributed By :



meetgor

@meetgor

Vote for difficulty

Easy

Normal

Medium

Hard

Expert

Article Tags : [Bash-Script](#), [Picked](#), [Linux-Unix](#)

Improve Article

Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !



A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

[About Us](#)
[Careers](#)
[In Media](#)
[Contact Us](#)
[Privacy Policy](#)
[Copyright Policy](#)

News

[Top News](#)
[Technology](#)
[Work & Career](#)
[Business](#)
[Finance](#)
[Lifestyle](#)
[Knowledge](#)

Web Development

Learn

[Algorithms](#)
[Data Structures](#)
[SDE Cheat Sheet](#)
[Machine learning](#)
[CS Subjects](#)
[Video Tutorials](#)
[Courses](#)

Languages

[Python](#)
[Java](#)
[CPP](#)
[Golang](#)
[C#](#)
[SQL](#)
[Kotlin](#)

Contribute

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

NodeJS

@geeksforgeeks , Some rights reserved

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !