Evaluate and execute code

Internal reference: topics/05-4.md

Introduction

The `eval` statement is a built-in command in Bash that allows you to dynamically evaluate and execute code that is either constructed or stored as a string.

The `eval` statement is frequently used when you need to generate and execute commands dynamically based on certain conditions or inputs. It enables you to construct complex commands on the fly, making use of variables, conditionals, loops, and other logic in the execution process.

Basic usage

To explore the basic usage, run the following command in your terminal window:

```
$ USER=Fritz
$ eval echo Hello $USER!
```

This will output the following:

```
Hello Fritz!
```

In this example, the following occurs:

1. You tell the `eval` statement that you want to output the word "Hello" as well as the name of the currently logged-in user, using the environment variable $USER.
2. The `eval` statement evaluates the string, recognizes that it contains an environment variable called $USER, and retrieves the value for that variable.
3. The output you receive confirms that it worked and your terminal window dutifully echoes Hello Fritz! back to you.

Advanced usage

Because the eval statement evaluates and detects commands and other variables from the string, you can end up with the same eval commands having different values.

For example, you might want to know what the day was exactly a week ago:

```
$ lastweek='date --date=1 week ago'
$ eval $lastweek
```

This will output the following:

```
Sat, Jan  6, 2024  8:53:19 PM
```

Run the same command a few seconds later, and the time value of $lastweek increases:

```
$ eval $lastweek
Sat, Jan  6, 2024  8:53:23 PM
```

This is because the eval statement runs the command again, which, by default, returns the latest value of date

```
--date=1 week ago
```

Daniel Garavaldi

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m122_aws/topics/05_4

Last update: **2025/10/30 23:06**