How to Write a Bash Script with Examples

December 23, 2021

BASH LINUX

Home » DevOps and Development » How to Write a Bash Script with Examples

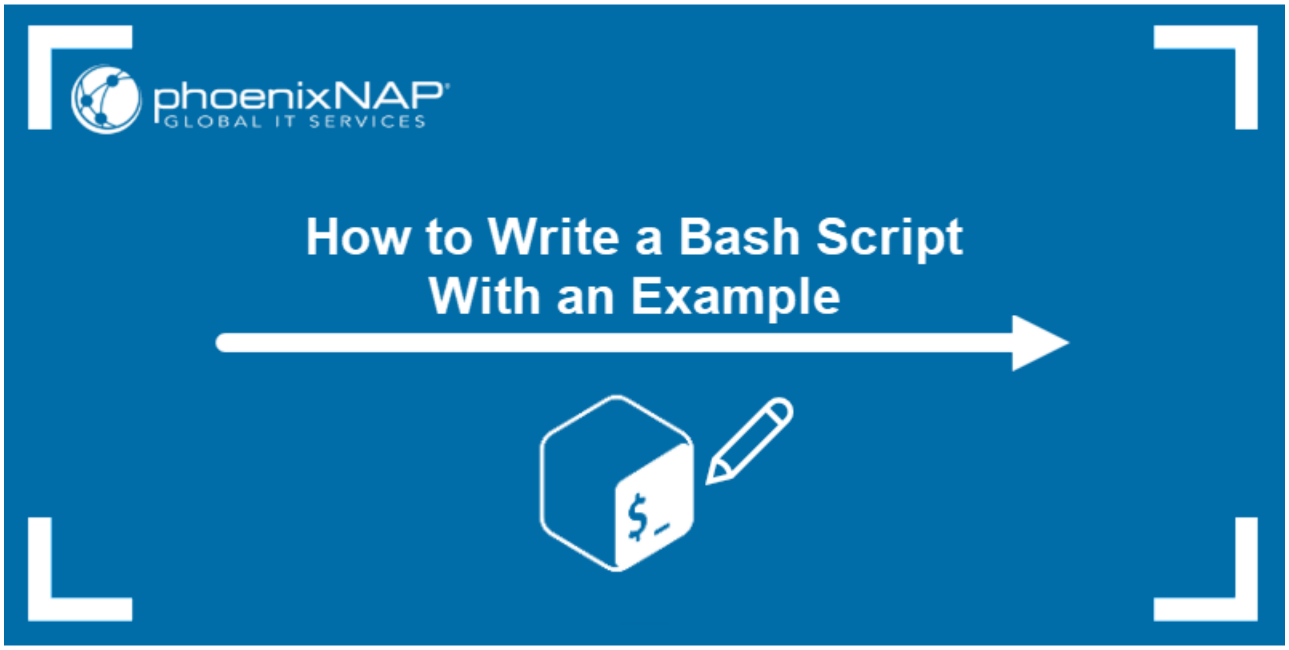
Contents

Introduction

Bash is a Unix command line interface for interacting with the operating system, available for Linux and macOS. Bash scripts help group commands to create a program. All instructions that run from the terminal work in Bash scripts as well.

Bash scripting is a crucial tool for system administrators and developers. Scripting helps automate repetitive tasks and interact with the OS through custom instruction combinations. The skill is simple to learn and requires only basic terminal commands to get started.

This guide will show how to write a Bash script and Bash scripting basics through examples.


phoenixNAP GLOBAL IT SERVICES
How to Write a Bash Script With an Example
$_

Prerequisites

- Access to the terminal (CTRL+ALT+T) with sudo privileges.
- Basic Linux commands (grab our Linux commands cheat sheet).
- A text editor, such Vi/Vim.

Writing a Bash Script

To start with Bash scripting, create a new file using a text editor. If you're using Vim, run the following command:

```
vim script.sh
```

The extension for Bash scripts is *.sh*. However, the extension is not necessary. Adding the *.sh* makes the file easy to identify and maintain.

Adding the "shebang"

The first line in Bash scripts is a character sequence known as the "shebang." The shebang is the program loader's first instruction when executing the file, and the characters indicate which interpreter to run when reading the script.

Add the following line to the file to indicate the use of the Bash interpreter:

```
#!/bin/bash
```

The shebang consists of the following elements:

- **`#!`** directs the program loader to load an interpreter for the code in the file.
- **`/bin/bash`** the Bash interpreter's location.

Some typical shebang lines for different interpreters are in the table below.

Shebang	Interpreter
`#!/bin/bash`	Bash
`#!/bin/sh`	Bourne shell
`#!/usr/bin/env <interpreter>`	Uses the **`env`** program to locate the interpreter. Use this shebang for other scripting languages, such as Perl, Python, etc.
`#!/usr/bin/pwsh`	Powershell

After adding a shebang, continue to the next section.

Note: Learn how to evaluate arithmetic expressions using Bash let.

Adding Comments

Comments are lines that do not execute. However, they help with code readability. After the shebang, add a comment to explain what the script is.

For example:

```
#!/bin/bash
# A simple Bash script
```

For more information on Bash comments and the best practices, read our article on how to comment in Bash.

Adding Code

As an example, create a **script to update and upgrade the system**. Add the lines after the Bash comment so the final script looks like the following:

```
#!/bin/bash
# A simple Bash script
sudo apt update -y
sudo apt upgrade -y
echo Done!
```

The Bash interpreter reads each line and executes the update command followed by the upgrade command. The **-y** tag automatically answers **Yes** to any prompt brought up by the two instructions. When completed, the program prints **Done!** to the console.

Save the code and exit the text editor.

Executing the Bash Script

To run the Bash script, use the following command in the terminal:

```
bash script.sh
```

The script prompts to enter the password to run the sudo commands. Enter the password and wait for the program to finish the update and upgrade.

Note: Learn how to use the Bash read command.

Conclusion

By following the steps in this tutorial, you should have a simple script to update and upgrade the system. Customize the script further or create a new script that does something different. Our guide could help you to start creating custom bash functions which could help you write more efficient bash scripts.

To further deepen your Bash knowledge, learn how to write a script to check if a file or directory exists in Bash.

Read our article Bash math operations (Bash arithmetic) and learn about different bash math commands and methods, and why is math important in Bash scripting.

Milica Dancuk

Milica Dancuk is a technical writer at phoenixNAP who is passionate about programming. Her background in Electrical Engineering and Computing combined with her teaching experience give her the ability to easily explain complex technical concepts through her

Was this article helpful? Yes No

content.

Next you should read

DevOps and Development, SysAdmin

Bash case Statement Syntax and Examples

December 15, 2021

The bash case statement simplifies complex conditional statements and tests...

READ MORE

DevOps and Development, SysAdmin

Bash Script for Loop Explained with Examples

December 15, 2021

This article shows how to use the for loop in bash scripts through various hands-on examples.

READ MORE

DevOps and

D
ev
el
o
p
m
en
t,
Sy
s
A
d
m

DevOps and Development

Bash Function & How to Use It {Variables, Arguments, Return}

November 3, 2021

Learn how to use functions in Bash scripting through this hands-on tutorial with example codes.

READ MORE

t
e
m
e
n

Live Chat Get a Quote Support | 1-855-330-1509 Sales | 1-877-588-5918

t: A Comprehensive Tutorial

October 21, 2021

Conditi

Contact Us
Legal
Privacy Policy
Terms of Use
DMCA
GDPR
Sitemap

o
n
al
st
at
e
m
en
ts
he
lp
a
ut
o
m
at
e
d
ec
isi
o
n
m
ak
in
g
pr
o
ce
ss
es
in.
..

**RE
A
D
M**

O

RE