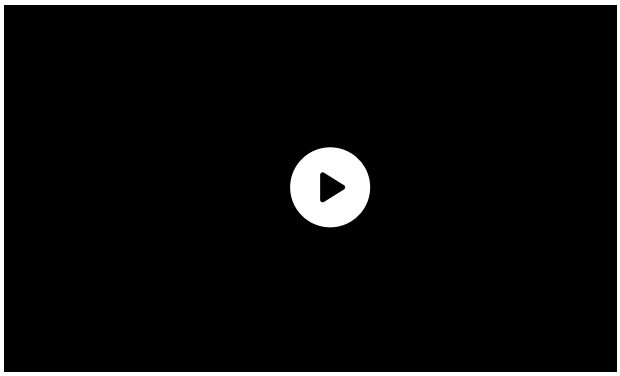
How to Use a Regex Inside an if Clause in Bash



Last updated: March 18, 2024

Written by: baeldung
(https://www.baeldung.com/linux/author/baeldung)

Reviewed by: Grzegorz Piwowarek
(https://www.baeldung.com/linux/editor/grzegorz-author)

Scripting (https://www.baeldung.com/linux/category/scripting)

bash (https://www.baeldung.com/linux/tag/bash)

regex (https://www.baeldung.com/linux/tag/regex)

1. Introduction

Regular expressions or regexes are a powerful tool for pattern matching in text data. Bash, being a command-line shell and programming language, has built-in support for regexes through its pattern-matching operators.

In Bash, people often use regexes in if statements to check whether a pattern matches a string. In this article, we'll demonstrate how to use a regex in an if clause in Bash.

2. Using a Regex Inside an *if* Clause

In Bash, we can use the `=~` operator to match a string against a regex pattern:

```
if [[ "$string" =~ regex_pattern ]]; then
  # code block to execute if string matches regex pattern
else
  # code block to execute if string does not match regex pattern
fi
```

The code tests a string variable named *$string* against a regex pattern called *regex_pattern*. The if statement is enclosed in double square brackets *[[ ]]*, and the *=~* operator checks if the string matches the regex pattern.

Let's take a look at a simple example to understand this better. Suppose we have a variable called *filename* containing the name of a file, and we want to check if it has the *.txt* extension.

We can use the =~ operator to match the filename against the regex pattern *.txt$*. The dollar sign ($) at the end of the pattern is used to anchor the pattern to the end of the string, ensuring that it matches only if the string ends with *.txt*:

```
filename="document.txt"

if [[ "$filename" =~ \.txt$ ]]; then
  echo "Filename has a .txt extension"
else
  echo "Filename does not have a .txt extension"
fi
```

In the above example, the if statement tests if the variable *filename* matches the regex pattern *.txt$*. Since the filename contains *.txt* at the end, the condition evaluates to true, and the output is *Filename has a .txt extension*.

3. Using a Negated Regex

Sometimes, we may want to check if a string does not match a particular pattern. In such cases, we can use the negation operator (!) before the =~ operator:

```
string="Hello, world!"

if [[ "$string" =~ [0-9] ]]; then
  echo "String does not contain any digits"
else
  echo "String contains at least one digit"
fi
```


In the above example, the if statement tests if the variable *string* does not match the regex pattern [0-9], which matches any digit. Since the string does not contain any digits, the condition evaluates to true, and the output is *String does not contain any digits*.

4. Using Regex with Variables

We can also use variables containing regex patterns in if statements:

```
pattern="[a-z]+"

if [[ "hello" =~ $pattern ]]; then
  echo "String matches the regex pattern"
else
  echo "String does not match the regex pattern"
fi
```

5. Conclusion

Bash's if clause can match text patterns with regex using =~ and double square brackets [[ ]]. Negated regexes can also be used to check for non-matching patterns. Regex patterns can be stored in variables for use in if statements. Furthermore, these features make regex a powerful tool for pattern matching in Bash.

In summary, regexes can greatly enhance the functionality of Bash scripts and command-line tools. By using them inside if clauses, we can create more powerful and flexible programs that can handle various text data.

Comments are open for 30 days after publishing a post. For any issues past this date, use the Contact form on the site.

CATEGORIES

ADMINISTRATION (/LINUX/CATEGORY/ADMINISTRATION)
FILES (/LINUX/CATEGORY/FILES)
FILESYSTEMS (/LINUX/CATEGORY/FILESYSTEMS)
INSTALLATION (/LINUX/CATEGORY/INSTALLATION)
NETWORKING (/LINUX/CATEGORY/NETWORKING)
PROCESSES (/LINUX/CATEGORY/PROCESSES)
SCRIPTING (/LINUX/CATEGORY/SCRIPTING)
SEARCH (/LINUX/CATEGORY/SEARCH)

SECURITY (/LINUX/CATEGORY/SECURITY)
WEB (/LINUX/CATEGORY/WEB)

(/linux/) (https://www.baeldung.com/linux/)

SERIES (/bael-search)

LINUX ADMINISTRATION (/LINUX/LINUX-ADMINISTRATION-SERIES)
LINUX FILES (/LINUX/LINUX-FILES-SERIES)
LINUX PROCESSES (/LINUX/LINUX-PROCESSES-GUIDE)
LINUX SECURITY TUTORIALS (HTTPS://WWW.BAELDUNG.COM/LINUX/LINUX-SECURITY-SERIES)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE FULL ARCHIVE (HTTPS://WWW.BAELDUNG.COM/LINUX/FULL_ARCHIVE)
EDITORS (HTTPS://WWW.BAELDUNG.COM/EDITORS)

TERMS OF SERVICE (HTTPS://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
PRIVACY POLICY (HTTPS://WWW.BAELDUNG.COM/PRIVACY-POLICY)
COMPANY INFO (HTTPS://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)
CONTACT (/CONTACT)