Linux | DB | Open Source | Web

Boost your career with Simplilearn's
Artificial Intelligence Engineer
Master's Program and Job Assist.

≡ Menu

Bash String Manipulation Examples – Length, Substring, Find and Replace

by Sasikala on July 23, 2010

Like 1 Tweet

In bash shell, when you use a dollar sign followed by a variable name, shell expands the variable with its value. This feature of shell is called parameter expansion.

But parameter expansion has numerous other forms which allow you to expand a parameter and modify the value or substitute other values in the expansion process. In this article, let us review how to use the parameter expansion concept for string manipulation operations.

This article is part of the on-going bash tutorial series. Refer to our earlier article on bash { } expansion.

1. Identify String Length inside Bash Shell Script

```
${#string}
```

The above format is used to get the length of the given bash variable.

```
$ cat len.sh
#! /bin/bash

var="Welcome to the geekstuff"

echo ${#var}

$ ./len.sh
24
```

To understand more about bash variables, read 6 Practical Bash Global and Local Variable Examples.

2. Extract a Substring from a Variable inside Bash Shell Script

Bash provides a way to extract a substring from a string. The following example expains how to parse n characters starting from a particular position.

```
${string:position}
```

Extract substring from $string at $position

```
${string:position:length}
```

Extract $length of characters substring from $string starting from $position. In the below example, first echo statement returns the substring starting from 15th position. Second echo statement returns the 4 characters starting from 15th position. Length must be the number greater than or equal to zero.

```
$ cat substr.sh
#! /bin/bash

var="Welcome to the geekstuff"

echo ${var:15}
echo ${var:15:4}

$ ./substr.sh
geekstuff
geek
```

Also, refer to our earlier article to understand more about [$*, $@, $#, $$, $!, $?, $-, $_ bash special parameters](#).

3. Shortest Substring Match

Following syntax deletes the shortest match of $substring from front of $string

```
${string#substring}
```

Following syntax deletes the shortest match of $substring from back of $string

```
${string%substring}
```

Following sample shell script explains the above two shortest substring match concepts.

```
$ cat shortest.sh
#! /bin/bash

filename="bash.string.txt"

echo ${filename#*.}
echo ${filename%.*}

$ ./shortest.sh
After deletion of shortest match from front: string.txt
After deletion of shortest match from back: bash.string
```

In the first echo statement substring '*.' matches the characters and a dot, and # strips from the front of the string, so it strips the substring "bash." from the variable called filename. In second echo statement substring '.*' matches the substring starts with dot, and % strips from back of the string, so it deletes the substring '.txt'

4. Longest Substring Match

Following syntax deletes the longest match of $substring from front of $string

```
${string##substring}
```

Following syntax deletes the longest match of $substring from back of $string

```
${string%%substring}
```

Following sample shell script explains the above two longest substring match concepts.

```
$ cat longest.sh
#! /bin/bash

filename="bash.string.txt"

echo "After deletion of longest match from front:" ${filename##*.}
echo "After deletion of longest match from back:" ${filename%%.*}

$ ./longest.sh
After deletion of longest match from front: txt
After deletion of longest match from back: bash
```

In the above example, ##*. strips longest match for '*.' which matches "bash.string." so after striping this, it prints the remaining txt. And %%.* strips the longest match for .* from back which matches ".string.txt", after striping it returns "bash".

5. Find and Replace String Values inside Bash Shell Script

Replace only first match

```
${string/pattern/replacement}
```

It matches the pattern in the variable $string, and replace only the first match of the pattern with the replacement.

```
$ cat firstmatch.sh
#! /bin/bash

filename="bash.string.txt"

echo "After Replacement:" ${filename/str*./operations.}

$ ./firstmatch.sh
After Replacement: bash.operations.txt
```

Replace all the matches

```
${string//pattern/replacement}
```

It replaces all the matches of pattern with replacement.

```
$ cat allmatch.sh
#! /bin/bash

filename="Path of the bash is /bin/bash"

echo "After Replacement:" ${filename//bash/sh}
```

```
$ ./allmatch.sh
After Replacement: Path of the sh is /bin/sh
```

Taking about find and replace, refer to our earlier articles – sed substitute examples and Vim find and replace.

Replace beginning and end

```
${string/#pattern/replacement}
```

Following syntax replaces with the replacement string, only when the pattern matches beginning of the $string.

```
${string/%pattern/replacement}
```

Following syntax replaces with the replacement string, only when the pattern matches at the end of the given $string.

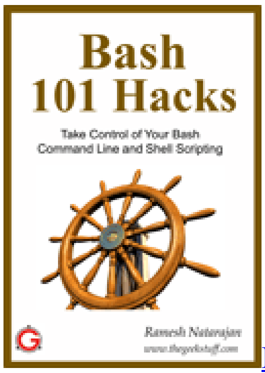
```
$ cat posmatch.sh
#! /bin/bash

filename="/root/admin/monitoring/process.sh"

echo "Replaced at the beginning:" ${filename/#\/root/\/tmp}
echo "Replaced at the end": ${filename/%.*/.ksh}

$ ./posmatch.sh
Replaced at the beginning: /tmp/admin/monitoring/process.sh
Replaced at the end: /root/admin/monitoring/process.ksh
```

Recommended Reading



Bash 101 Hacks, by Ramesh Natarajan. I spend most of my time on Linux environment. So, naturally I'm a huge fan of Bash command line and shell scripting. 15 years back, when I was working on different flavors of *nix, I used to write lot of code on C shell and Korn shell. Later years, when I started working on Linux as system administrator, I pretty much automated every possible task using Bash shell scripting. Based on my Bash experience, I've written Bash 101 Hacks eBook that contains 101 practical examples on both Bash command line and shell scripting. If you've been thinking about mastering Bash, do yourself a favor and read this book, which will help you take control of your Bash command line and shell scripting.

Tweet Like 1 > Add your comment

If you enjoyed this article, you might also like..

1. 50 Linux Sysadmin Tutorials
2. 50 Most Frequently Used Linux Commands (With Examples)
3. Top 25 Best Linux Performance Monitoring and Debugging Tools
4. Mommy, I found it! – 15 Practical Linux Find Command Examples

- Awk Introduction – 7 Awk Print Examples
- Advanced Sed Substitution Examples
- 8 Essential Vim Editor Navigation Fundamentals
- 25 Most Frequently Used Linux IPTables Rules Examples