Bash Split String

Bash Split String – Often when working with string literals or message streams, we come across a necessity to split a string into tokens using a delimiter. The delimiter could be a single character or a string with multiple characters. In this tutorial, we shall learn how to split a string in bash shell scripting with a delimiter of single and multiple character lengths.

Bash Split String

Split String with single character delimiter(s) in Bash using IFS

To split a string in bash using IFS, follow the below steps:

Step 1: Set IFS to the delimiter you would want. `IFS='<delimiter>'` IFS is an internal variable that determines how Bash recognizes word boundaries. The default value of IFS is white space. If you set it to some other value, reset it to default whitespace.

Step 2: Read your string to a variable with options -ra. `read -ra ARR <<< "$str"`

Option	Description
-r	Backslash does not act as an escape character.
-a ARR	The words(separated by IFS) are assigned to the sequential index of array ARR beginning at zero.

Now you have your string split by the delimiter (set in IFS) stored in array ARR. ARR is just array name. Any string literal that is valid could be used as an array name.

Step 3: You may now access the tokens split into an array using a bash for loop.

Its time for some examples. In the following two examples, we will go through example bash scripts where we use IFS to split a string.

Example 1: Bash Split String by Space

In this example, we split a string using space as a character delimiter.

Bash Script File

```
#!/bin/bash

str="Learn to Split a String in Bash Scripting"

IFS=' '     # space is set as delimiter
read -ra ADDR <<< "$str"   # str is read into an array as tokens separated by IFS
for i in "${ADDR[@]}"; do   # access each element of array
    echo "$i"
done
```

Run the above bash shell script in a terminal.

Output

```
$ ./bash-split-string-example
Learn
to
Split
a
String
in
Bash
Scripting
```

Example 2: Bash Split String by Symbol

Sometimes we may need to split a string by a delimiter other than space. To split a string in bash shell by a symbol or any other character, set the symbol or specific character to IFS and read the string to a variable with the options `-ra` mentioned in the below example.

Bash Script File

```
#!/bin/bash

str="Learn-to-Split-a-String-in-Bash-Scripting"

IFS='-'     # hyphen (-) is set as delimiter
read -ra ADDR <<< "$str"   # str is read into an array as tokens separated by IFS
for i in "${ADDR[@]}"; do   # access each element of array
    echo "$i"
done
IFS=' '     # reset to default value after usage
```

Run the above bash shell script in terminal.

Output

```
$ ./bash-split-string-example-1
```

```
Learn
to
Split
a
String
in
Bash
Scripting
```

The default value of IFS is single space `' '`. We changed the value of IFS to split a string. It should not affect any further script that is dependent on IFS. So, assign the default value back to IFS.

Split String with multiple character delimiter

To split a string with a multiple character delimiter (or simply said another string), following are two of the many possible ways, one with idiomatic and the other with just basic bash if and bash while loop.

Example 3: Split String with another string as delimiter idiomatic expressions

In this example, we will use idiomatic expressions where parameter expansion is done, and thus a very compact script.

Bash Script File

```
#!/bin/bash

str="LearnABCtoABCSplitABCaABCString"
delimiter=ABC
s=$str$delimiter
array=();
while [[ $s ]]; do
    array+=( "${s%%"$delimiter"*}" );
    s=${s#*"$delimiter"};
done;
declare -p array
```

When you run the above script in a Bash Shell, you will see an output similar to the following

Output

```
$ ./bash-split-string-example-3
declare -a array='([0]="Learn" [1]="to" [2]="Split" [3]="a" [4]="String")'
```

Following Parameter-Expansions are used (Reference: Bash Man Page[https://linux.die.net/man/1/bash])

Expansion	Description
${parameter%%word}	Remove the longest matching suffix pattern.

${parameter%%word}	Remove the longest matching suffix pattern.
${parameter#word}	Remove shortest matching prefix pattern.

Example 4: Bash Split a String with multiple character delimiter

If you are new to bash shell scripting and are not familiar with idiomatic expressions, following is an easily understandable shell script with basic bash if, bash while and bash substring methods. Comments are provided at each step to make the script file more readable.

Bash Script File

```
#!/bin/bash

# main string
str="LearnABCtoABCSplitABCaABCStringABCinABCBashABCScripting"

# delimiter string
delimiter="ABC"

#length of main string
strLen=${#str}
#length of delimiter string
dLen=${#delimiter}

#iterator for length of string
i=0
#length tracker for ongoing substring
wordLen=0
#starting position for ongoing substring
strP=0

array=()
while [ $i -lt $strLen ]; do
    if [ $delimiter == ${str:$i:$dLen} ]; then
        array+=(${str:strP:$wordLen})
        strP=$(( i + dLen ))
        wordLen=0
        i=$(( i + dLen ))
    fi
    i=$(( i + 1 ))
    wordLen=$(( wordLen + 1 ))
done
array+=(${str:strP:$wordLen})

declare -p array
```

Output

```
$ ./bash-split-string-example
declare -a array='([0]="Learn" [1]="to" [2]="Split" [3]="a" [4]="String" [5]="in" [6]="B
```

The split strings are stored in the array and could be accessed using an index.

Conclusion

In this Bash Tutorial, we have learned how to split a string using bash script with different scenarios based on delimiter: like single character delimiter and multiple character delimiter.

Bash Shell Scripting

- Bash Tutorial
- Bash Script Example
- Bash File Extension
- Bash Echo
- Bash Comments
- Bash Variable
- Bash Command Line Arguments
- Bash Read User Input
- Bash Read Password
- Bash Date Format
- Bash Sleep

Operators

- Bash Arithmetic Operators

Conditional Statements

- Bash If
- Bash If Else
- Bash Else If
- Bash Case

Loops

- Bash For Loop
- Bash While Loop
- Bash Until Loop

Strings

- Bash String Manipulation Examples
- Bash String Length
- Bash If String Equals
- **Bash Split String**
- Bash SubString
- Bash Concatenate String
- Bash Concatenate Variables to Strings

Functions

- Bash Function
- Bash Override Buitlin Commands

Arrays

- Bash Array

Files

- Bash Write to File
- Bash Read File
- Bash Read File line by line
- Bash If File Exists
- Bash If File is Directory
- Bash If File is Readable

Bash Others

- Bash Check if variable is set