Bash Split String

In this topic, we have defined how to split a string in bash shell scripting.

In some cases, we might need to split the string data to perform some specific tasks. Most of the programming languages contain built-in function 'split' to divide any string data into multiple parts. However, bash does not contain such type of built-in function. But we can use delimiters to split any string data in bash scripting. The delimiter can be either a single character or a string with multiple characters.

Check out the methods below to understand how to split string in a bash shell:

Split using $IFS variable

Following are the steps to split a string in bash using $IFS:

- $IFS is a special internal variable which is used to split a string into words. $IFS variable is called '**Internal Field Separator**' which determines how Bash recognizes boundaries. $IFS is used to assign the specific delimiter [**IFS=''**] for dividing the string. The white space is a default value of $IFS. However, we can also use values such as '\t', '\n', '-' etc. as the delimiter.
- After assigning the delimiter, a string can be read by two options: '-r' and '-a'. i.e., **read -ra ARR <<< "$str"**.
  Here, the option '-r' is used to define that backslash (\) is a character rather than escape character. The '-a' option is used to define that the words (separated by $IFS) are assigned to the sequential index of array beginning at zero.
- Then we apply bash 'for' loop to access the tokens which are split into an array.

Let's understand this mechanism with the help of some examples:

Example 1: Bash Split String by Space

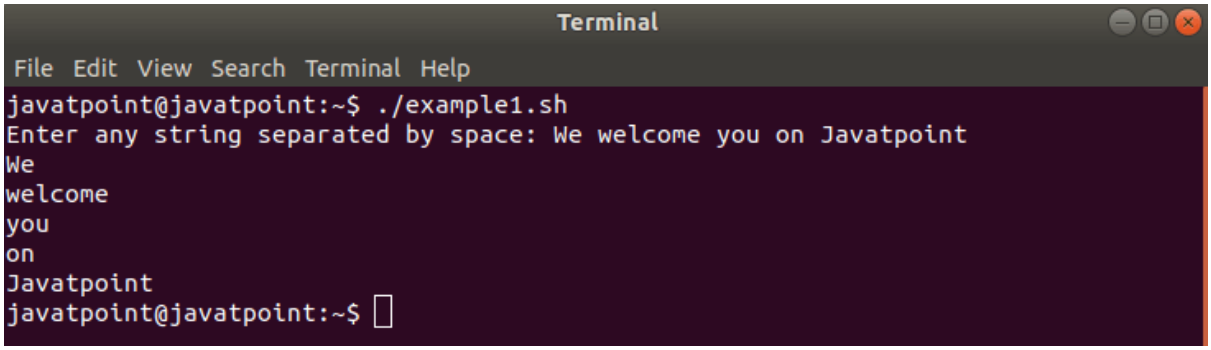
In this example, a string is split using a space character delimiter.

Bash Script

```
#!/bin/bash
#Example for bash split string by space
read -p "Enter any string separated by space: " str  #reading string value
IFS='' #setting space as delimiter
read -ra ADDR <<<"$str" #reading str as an array as tokens separated by IFS
for i in "${ADDR[@]}"; #accessing each element of array
do
echo "$i"
done
```

Output

If we input a string "We welcome you on Javatpoint", the output will look like this:



Example 2: Bash Split String by Symbol

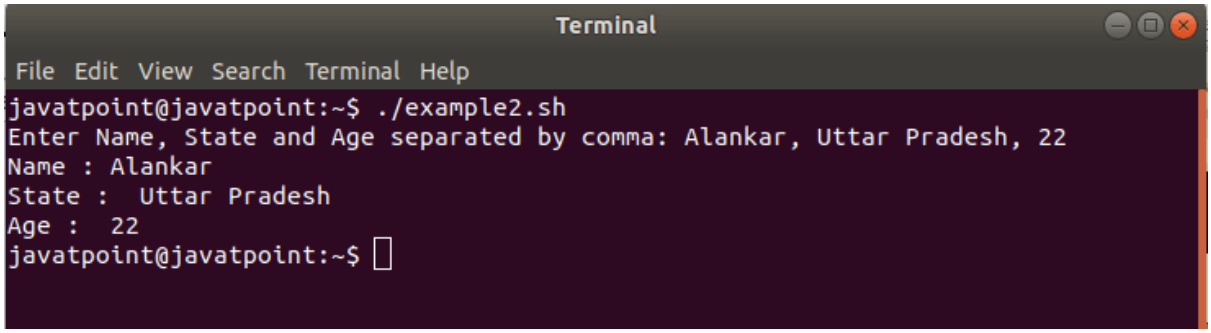
In some cases, we may have a requirement to split a string by other delimiters such as a symbol or specific character. In this example, a string is split using a comma (,) symbol character as a delimiter.

Bash Script

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

```
#Example for bash split string by Symbol (comma)

read -
p "Enter Name, State and Age separated by a comma: " entry #reading string value

IFS=',' #setting comma as delimiter

read -a strarr <<<"$entry" #reading str as an array as tokens separated by IFS

echo "Name : ${strarr[0]} "

echo "State : ${strarr[1]} "

echo "Age : ${strarr[2]}"
```

Output



Split without $IFS variable

In bash, a string can also be divided without using $IFS variable. The 'readarray' command with -d option is used to split the string data. The -d option is applied to define the separator character in the command like $IFS. Moreover, the bash loop is used to print the string in split form.

Let's understand this logic with the help of some example:

Example 1: Bash Split String by Symbol

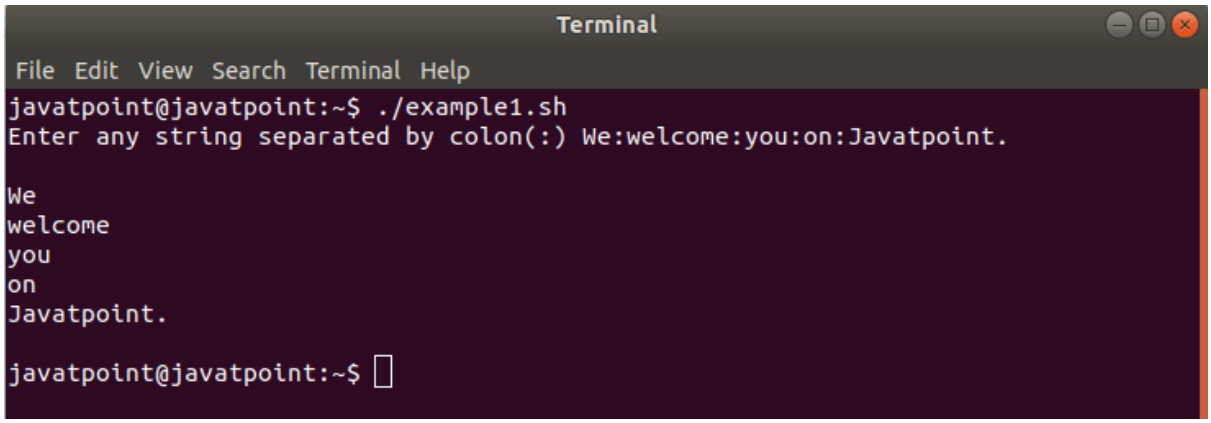
This example defines how a string value can be split without using $IFS. As per the script, a text value should be entered with the colon (:) sign so that it can be split. Check out the bash script below:

Bash Script

```
#!/bin/bash

#Example for bash split string without $IFS
```

```
read -p "Enter any string separated by colon(:) " str #reading string value
readarray -d : -t strarr <<<"$str" #split a string based on the delimiter ':'
printf "\n"
#Print each value of Array with the help of loop
for (( n=0; n < ${#strarr[*]}; n++ ))
do
echo "${strarr[n]}"
done
```

Output



Example 2: Bash Split String by another string

In this example, we have used idiomatic expressions where parameter expansion has completed.

Bash Script

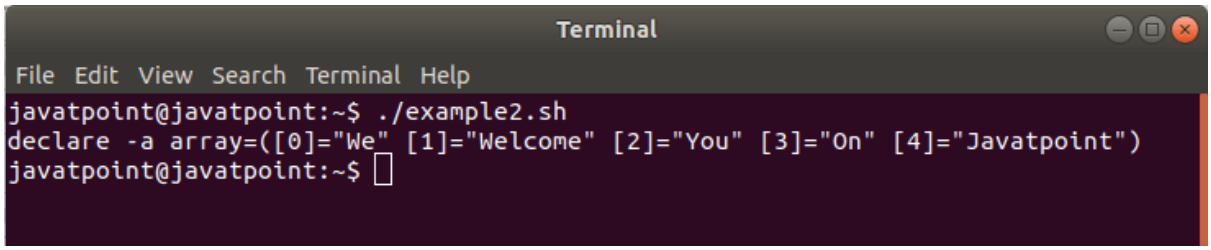
```
#!/bin/bash
#Example for bash split string by another string

str="WeLearnWelcomeLearnYouLearnOnLearnJavatpoint"
delimiter=Learn
s=$str$delimiter
array=();
while [[ $s ]];
```

```
do
array+=( "${s%%"$delimiter"*}" );
s=${s#*"$delimiter"};
done;
declare -p array
```

In this bash script, we have used the following Parameter-Expansions:

- **${parameter%%word}**
  It removes the longest matching suffix pattern.
- **${parameter#word}**
  It removes the shortest matching prefix pattern.

Output



Example 3: Bash Split String using Trim Command

In this example, we have used trim (tr) command to split a string. Instead of using the read command, the trim command is used to split a string on the delimiter.

Bash Script

```
#!/bin/bash
#Example to split a string using trim (tr) command
my_str="We;welcome;you;on;javatpoint."
my_arr=($(echo $my_str | tr ";""\n"))
for i in "${my_arr[@]}"
```

```
do
echo $i
done
```

Output