Bash Array – Declare, Initialize and Access – Examples

Bash Array

Bash Array – An array is a collection of elements. Unlike in many other programming languages, in bash, an array is not a collection of similar elements. Since bash does not discriminate string from a number, an array can contain a mix of strings and numbers.

In this Bash Tutorial, we shall learn how to declare, initialize and access one dimensional Bash Array, with the help of examples. Also, we shall look into some of the operations on arrays like appending, slicing, finding the array length, etc.

Bash Array Declaration

To declare a variable as a Bash Array, use the keyword `declare` and the syntax is

```
declare -a ARRAY_NAME
```

where

ARRAY_NAME	Name that you would give to the array. Rules of naming a variable in bash hold for naming array as well

Bash Array Initialization

To initialize a Bash Array, use assignment operator `=`, and enclose all the elements inside braces `()`. The syntax to initialize a bash array is

```
ARRAY_NAME=( ELEMENT_1 ELEMENT_2 ELEMENT_N )
```

ELEMENT_N	Nth element of array	

Note that there has to be no space around the assignment operator `=`.

Access elements of Bash Array

You can access elements of a Bash Array using the index.

```
echo ${ARRAY_NAME[2]}
```

Print Bash Array with all the information

To print all the elements of a bash array with all the index and details use `declare` with option `p`. They syntax to print the bash array is

```
declare -p ARRAY_NAME
```

Examples

Enough with the syntax and details, let's see bash arrays in action with the help of these example scripts.

Example 1: Bash Array

Following is an example Bash Script in which we shall create an array `names`, initialize it, access elements of it and display all the elements of it.

Bash Shell Script

```
#!/bin/bash

# declare names as an indexed array
declare -a names

# initialize the array
names=( Miller Ted Susan Gary )

# access elements of array using index
echo ${names[2]}

# you may display the atrributes and value of each element of array
```

```
declare -p names
```

Output

```
~$ ./bash-array-example
Susan
declare -a names='([0]="Miller" [1]="Ted" [2]="Susan" [3]="Gary")'
```

Example 2: Accessing elements of Array using For Loop

In this example, we shall learn to access elements of an array iteratively using Bash For Loop.

Bash Shell Script

```
#!/bin/bash

# declare an array
arr=( "bash" "shell" "script" )

# for loop that iterates over each element in arr
for i in "${arr[@]}"
do
    echo $i
done
```

Output

```
~$ ./bash-array-example-2
bash
shell
script
```

Append Elements to Bash Array

To append an element to array in bash, use `+=` operator and element enclosed in parenthesis.

Following is an example to demonstrate how to append an element to array.

Bash Shell Script

```
#!/bin/bash

arr=( "bash" "shell" "script" )

arr+=("tutorial")

# for loop that iterates over each element in arr
for i in "${arr[@]}"
```

```
do
    echo $i
done
```

Output

```
~$ ./bash-array-example-2
bash
shell
script
tutorial
```

You can append multiple elements by providing them in the parenthesis separated by space. It is like appending another array to the existing array.

Note: If you miss parenthesis while appending, the element is not added to the array, but to the first element of the array.

Bash Array Length

If `arr` is the array, use the syntax `${#arr[@]}` to calculate its length.

Bash Shell Script

```
#!/bin/bash

arr=( "bash" "shell" "script" )

echo ${#arr[@]}
```

Output

```
~$ ./bash-array-example-2
3
```

Slice Bash Array

Bash array could be sliced from a starting index to ending index.

To slice an array `arr` from starting index `m` to ending index `n`, use the syntax

Following is an example to slice an array.

Bash Shell Script

```
#!/bin/bash

arr=( "bash" "shell" "script" "tutorial" )

arr2=("${arr[@]:1:2}")

# for loop that iterates over each element in arr
for i in "${arr2[@]}"
do
    echo $i
done
```

Output

```
~$ ./bash-array-example-2
shell
script
```

Conclusion

Concluding this Bash Tutorial, we have learned how to declare, initialize and access one dimensional Bash Array, with the help of examples.

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