Array

April 14, 2024

1 Define Array in SS?

In Shell Scripting, an array is a variable that can hold multiple values. The values are indexed by integers, starting at 0 for the first element.

Here's how you can define an array and access its elements:

[ ]:
```
#!/bin/bash
# Define an array
fruits=("Apple" "Banana" "Cherry")

# Print the entire array
echo "All fruits: ${fruits[@]}"
echo "All fruits: ${fruits[*]}"

# Print the first element of the array
echo "First fruit: ${fruits[0]}"

# Print the second element of the array
echo "Second fruit: ${fruits[1]}"

# Print the third element of the array
echo "Third fruit: ${fruits[2]}"
```

In this script, `fruits` is an array that contains three elements: "Apple", "Banana", and "Cherry". The `${fruits[@]}` syntax is used to access all elements of the array, and the `${fruits[index]}` syntax is used to access the element at a specific index.

Note that arrays in Shell Scripting are zero-indexed, which means the first element is at index 0, the second element is at index 1, and so on.

2 How to find length of Array

In a shell script, you can find the length of an array (i.e., the number of elements in the array) using the `${#array[@]}` syntax. Here's how you can modify your script to print the length of the `fruits` array:

[ ]:
```
#!/bin/bash
# Define an array
```

```
fruits=("Apple" "Banana" "Cherry")

# Print the entire array
echo "All fruits: ${fruits[@]}"

# Print the length of the array
echo "Number of fruits: ${#fruits[@]}"

# Print the first element of the array
echo "First fruit: ${fruits[0]}"

# Print the second element of the array
echo "Second fruit: ${fruits[1]}"

# Print the third element of the array
echo "Third fruit: ${fruits[2]}"
```

In this script, `${#fruits[@]}` is used to get the length of the `fruits` array. This will print `Number of fruits: 3`, because there are three elements in the `fruits` array.

3 Slice In Array

In a shell script, you can slice an array (i.e., get a subset of the elements in the array) using the `${array[@]:start:length}` syntax. Here's how you can modify your script to print a slice of the `fruits` array:

[ ]:
```
#!/bin/bash
# Define an array
fruits=("Apple" "Banana" "Cherry" "Date" "Elderberry")

# Print the entire array
echo "All fruits: ${fruits[@]}"

# Print a slice of the array (from index 1, length 3)
echo "Some fruits: ${fruits[@]:1:3}"

# Print the first element of the array
echo "First fruit: ${fruits[0]}"

# Print the second element of the array
echo "Second fruit: ${fruits[1]}"

# Print the third element of the array
echo "Third fruit: ${fruits[2]}"

# Print the length of the array
echo "Number of fruits: ${#fruits[@]}"
```

In this script, `${fruits[@]:1:3}` is used to get a slice of the `fruits` array. This will print `Some fruits: Banana Cherry Date`, because it starts at index 1 (the second element, because arrays are zero-indexed) and includes 3 elements.

4 Key-Value Pair Array In SS

In Bash version 4 and later, you can create an associative array, which is an array that uses keys instead of numerical indices. Here's how you can define an associative array and access its elements:

[ ]:
```
#!/bin/bash
# Define an associative array
declare -A fruits
fruits["a"]="Apple"
fruits["b"]="Banana"
fruits["c"]="Cherry"

# Print the entire array
echo "All fruits: ${fruits[@]}"

# Print the value associated with the key "a"
echo "Fruit a: ${fruits[a]}"

# Print the value associated with the key "b"
echo "Fruit b: ${fruits[b]}"

# Print the value associated with the key "c"
echo "Fruit c: ${fruits[c]}"
```

In this script, `fruits` is an associative array that contains three key-value pairs. The `${fruits[@]}` syntax is used to access all values of the array, and the `${fruits[key]}` syntax is used to access the value associated with a specific key.

Note that associative arrays are not ordered, so the order in which the elements are printed may not be the same as the order in which they were defined.

5 How can I iterate over the elements of an array in a shell script?

In a shell script, you can iterate over the elements of an array using a `for` loop. Here's how you can modify your script to iterate over the elements of the `fruits` associative array:

[ ]:
```
#!/bin/bash
# Define an associative array
declare -A fruits
fruits["a"]="Apple"
fruits["b"]="Banana"
fruits["c"]="Cherry"

# Iterate over the keys of the array
```

```
for key in "${!fruits[@]}"; do
    # Print the key and the value associated with the key
    echo "Key: $key"
    echo "Value: ${fruits[$key]}"
done
```

In this script, the `for key in "${!fruits[@]}"; do ... done` loop iterates over the keys of the `fruits` associative array. For each key, it prints the key and the value associated with the key.

Note that `"${!fruits[@]}"` is used to get all keys of the array, and `${fruits[$key]}` is used to get the value associated with a specific key.

6 Thank You!