# Bash String Manipulation: Length, Substring

In shell, when we 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.

**Identify String Length Shell Script**

${#string}

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

### Extract a Substring from a Variable inside Shell Script

Bash provides a way to extract a substring from a string.

${string:position}

Extract substring from $string at $position

${string:position:length}

Extract $length of characters substring from $string starting from $position.

# Bash Conditional Expression

Bash expression is the combination of operators, features, or values used to form a bash conditional statement. Conditional expression could be binary or unary expression which involves numeric, string or any commands whose return status is zero when success.

There are several conditional expressions that could be used to test with the files. Following are few conditional expressions that are helpful.

* [ -e filepath ] Returns true if file exists.
* [ expr1 -a expr2 ] Returns true if both the expression is true.
* [ expr1 -o expr2 ] Returns true if either of the expression1 or 2 is true.

### Check File Existence

The following shell script gets the filename with its absolute path, and checks if the file exists or not and it throws the appropriate information.

#! /bin/bash

file=$1

if [ -e $file ]

then

echo -e "File $file exists"

else

echo -e "File $file doesnt exists"

fi

**Declaring an Array and Assigning values**

In bash, array is created automatically when a variable is used in the format like,

name[index]=value

* name is any name for an array
* index could be any number or expression that must evaluate to a number greater than or equal to zero.You can declare an explicit array using declare -a arrayname.
* To access an element from an array use curly brackets like ${name[index]}.

#! /bin/bash

Unix[0]='Debian'

Unix[1]='Red hat'

Unix[2]='Ubuntu'

Unix[3]='Suse'

echo ${Unix[1]}

### Initializing an array during declaration

Instead of initializing an each element of an array separately, we can declare and initialize an array by specifying the list of elements (separated by white space) with in a curly braces.

Syntax:

declare -a arrayname=(element1 element2 element3)

If the elements has the white space character, enclose it with in a quotes.

#! /bin/bash

declare -a Unix=('Debian' 'Red hat' 'Red hat' 'Suse' 'Fedora');

declare -a declares an array and all the elements in the parentheses are the elements of an array.

### Print the Whole Array

There are different ways to print the whole elements of the array. If the index number is @ or \*, all members of an array are referenced. We can traverse through the array elements and print it, using looping statements in bash.

echo ${Unix[@]}

Referring to the content of a member variable of an array without providing an index number is the same as referring to the content of the first element, the one referenced with index number zero.

### Length of the Bash Array

We can get the length of an array using the special parameter called $#.

${#arrayname[@]} gives the length of the array.

declare -a Unix=('Debian' 'Red hat' 'Suse' 'Fedora');

echo ${#Unix[@]} #Number of elements in the array

echo ${#Unix} #Number of characters in the first element of the array.i.e Debian

......

4

6

### Length of the nth Element in an Array

${#arrayname[n]} should give the length of the nth element in an array.

#! /bin/bash

Unix[0]='Debian'

Unix[1]='Red hat'

Unix[2]='Ubuntu'

Unix[3]='Suse'

echo ${#Unix[3]} # length of the element located at index 3 i.e Suse

....

4

### Extraction by offset and length for an array

The following example shows the way to extract 2 elements starting from the position 3 from an array called Unix.

$cat arraymanip.sh

Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux');

echo ${Unix[@]:3:2}

...

Suse Fedora

The above example returns the elements in the 3rd index and fourth index. Index always starts with zero.

### Extraction with offset and length, for a particular element of an array

To extract only first four elements from an array element . For example, Ubuntu which is located at the second index of an array, we can use offset and length for a particular element of an array.

#! /bin/bash

Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux');

echo ${Unix[2]:0:4}

..

Ubun

The above example extracts the first four characters from the 2nd indexed element of an array.