Birla Institute of Technology & Science, Pilani, K. K Birla Goa Campus
Computer Programming (CS F111)
Second Semester 2017-2018
Lab-04 (UNIX- Shell scripting)

Objectives

1. Introduction
2. Introduction to Arrays
3. Math Library Functions
4. Exercises

---

1. Introduction

In the last lab (lab 3) we learned how to write an interactive shell script programs using various loop constructs. In this lab we are going to focus on Arrays and Math library functions.

2. Introduction to Arrays

An array is a variable containing multiple values may be of same type . There is no maximum limit to the size of an array, nor any requirement that member variables be indexed or assigned contiguously. Array index starts with zero. To run all scripts related to arrays, use bash filename.

num[0]	num[1]	num[2]	num[3]	num[4]
2	8	7	6	0
Element1	Element2	Element3	Element4	Element5

Figure: An Example of array

2.1 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.

```
Unix[0]='Debian'
Unix[1]='Red hat'
Unix[2]='Ubuntu'
Unix[3]='Suse'
or

Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux');
echo ${Unix[1]}
```

output:
Red hat

To access an element from an array use curly brackets like ${name[index]}.

2.2 Print the Whole Bash 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. You can traverse through the array elements and print it, using looping statements in bash.

echo ${Unix[@]}
output:
Debian Red hat Ubuntu Suse

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 0.

2.3 Length of the Bash Array

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

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

2.4 Length of the nth Element in an Array

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

echo ${#Unix[3]} # length of the element located at index 3 i.e Suse
output:
4

2.5 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.

echo ${Unix[@]:3:2}
output:
Suse Fedora

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

2.6 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, you can use offset and length for a particular element of an array.

```
echo ${Unix[2]:0:4}
output:
Ubun
```

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

2.7 Search and Replace in an array elements

The following example, searches for Ubuntu in an array elements, and replace the same with the word ‘SCO Unix’.

```
echo ${Unix[@]/Ubuntu/SCO Unix}
output:
Debian Red hat SCO Unix Suse Fedora UTS OpenLinux
```

In this example, it replaces the element in the 2nd index ‘Ubuntu’ with ‘SCO Unix’. But this example will not permanently replace the array content.

2.8 Add an element to an existing Bash Array

The following example shows the way to add an element to the existing array.

```
Unix=("${Unix[@]}" "AIX" "HP-UX")
```

In the array called Unix, the elements ‘AIX’ and ‘HP-UX’ are added in 7th and 8th index respectively.

2.9 Remove an Element from an Array

unset is used to remove an element from an array.unset will have the same effect as assigning null to an element.

```
unset Unix[3]
echo ${Unix[3]}
```

The above script will just print null which is the value available in the 3rd index. The following example shows one of the way to remove an element completely from an array.

2.10 Remove Bash Array Elements using Patterns

In the search condition you can give the patterns, and stores the remaining element to an another array as shown below.

```
Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora');
patter=( ${Unix[@]/Red*/} )
echo ${patter[@]}
output:
Debian Ubuntu Suse Fedora
```

The above example removes the elements which has the patter Red*.

2.11 Copying an Array

Expand the array elements and store that into a new array as shown below.

```
Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux');
Linux=("${Unix[@]}")
echo ${Linux[@]}
output:
Debian Red hat Ubuntu Fedora UTS OpenLinux
```

2.12 Concatenation of two Bash Arrays

Expand the elements of the two arrays and assign it to the new array.

```
Unix=('Debian' 'Red hat' 'Ubuntu' 'Suse' 'Fedora' 'UTS' 'OpenLinux');
Shell=('bash' 'csh' 'jsh' 'rsh' 'ksh' 'rc' 'tcsh');
UnixShell=("${Unix[@]}" "${Shell[@]}")
echo ${UnixShell[@]}
echo ${#UnixShell[@]}
output:
Debian Red hat Ubuntu Suse Fedora UTS OpenLinux bash csh jsh rsh ksh rc tcsh
```

It prints the array which has the elements of the both the array ‘Unix’ and ‘Shell’, and number of elements of the new array is 14.

2.13 Deleting an Entire Array

unset is used to delete an entire array.

```
unset Unix
echo ${#UnixShell[@]}
output:
0
```

After unset an array, its length would be zero as shown above.

3. Math library functions

If **bc** is invoked with the **-l** option, a math library is preloaded and the default scale is set to 20. The math functions will calculate their results to the scale set at the time of their call. The math library defines the following functions:

s (*x*)
The sine of *x*, *x* is in radians.

c (*x*)
The cosine of *x*, *x* is in radians.

a (*x*)
The arctangent of *x*, arctangent returns radians.

l (*x*)
The natural logarithm of *x*.

e (*x*)
The exponential function of raising *e* to the value *x*.

Example
The following will assign the value of "pi" to the shell variable *pi*.
pi=$(echo "scale=10; 4*a(1)" | bc -l)
scale=10 means value of pi will have 10 digit precision.

Output:
3.1415926532

4. Exercises

1. Sort the given five integer numbers in ascending order (using array)

```
# Declare the array of 5 subscripts to hold 5 numbers
nos=(4 -1 2 66 10)
# Prints the number before sorting
echo "Original Numbers in array:"
for (( i = 0; i <= 4; i++ ))
do
  echo ${nos[$i]}
done
# Now do the Sorting of numbers
for (( i = 0; i <= 4 ; i++ ))
do
   for (( j = $i; j <= 4; j++ ))
   do
     if [ ${nos[$i]} -gt ${nos[$j]}  ]; then
         t=${nos[$i]}
         nos[$i]=${nos[$j]}
         nos[$j]=$t
     fi
   done
done
```

```
# Print the sorted number
echo -e "\nSorted Numbers in Ascending Order:"
for (( i=0; i <= 4; i++ ))
do
  echo ${nos[$i]}
done
```

Compulsory Question to Solve

Initialize an Array by taking data from a file [the file name is a command line argument]. Find Second largest number stored in the array [you are not allowed to sort the array].
[Hint: You can very well solve it with the help of one loop]