**Bash Scripting: Conditionals**

A conditional in [Bash scripting](https://linuxconfig.org/bash-scripting-tutorial-for-beginners) is made up of two things:

* conditional statement
* one or more conditional operators.

[Bash scripts](https://linuxconfig.org/bash-scripting-tutorial) give us two options for writing conditional statements.

* [if statement](https://linuxconfig.org/bash-if-statements-if-elif-else-then-fi)
* [case statement](https://linuxconfig.org/bash-script-case-statement-examples)

**Conditional Operators**

**A screenshot of a computer

Description automatically generated**

**A white sheet with black text

Description automatically generated**

***Loops in shell scripting***

* **A loop is a very useful programming tool that allows you to execute a set of command repeatedly.**
* **A loop is a programming structure that repeats a sequence of instructions until a specific condition is met.**

***Bash Scripting – While Loop***

* **The while loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.** **do while'*loop to execute a block of code repeatedly as long as a certain condition is true.***

Syntax of a while loop in BASH Scripting

**while [ condition ];  
do  
 # statements  
 # commands  
done**

***#!/bin/bash***

***a=7  
while [ $a -gt 4 ];  
do  
echo $a  
((a–))  
done  
echo “Out of the loop”***

* **#!/bin/bash: This line is called a shebang and indicates the path to the interpreter that should be used to execute the shell script.**
* **a=7**: Initializes a variable named a with the value 7.
* **while [ $a -gt 4 ];**: Starts a while loop that continues as long as the value of a is greater than 4.
* **do**: Marks the beginning of the code block to be executed within the while loop.
* **echo $a:** Prints the current value of the variable a to the console
* **((a--))**: Decrements the value of a by 1. This is a shorthand way of writing a=$((a - 1)).
* **done**: Marks the end of the code block for the while loop.
* **echo "Out of the loop"**: Prints “Out of the loop” to the console after the while loop has completed.

***Bash Scripting – for Loop***

* The for loop operates on lists of items. It repeats a set of commands for every item in a list.

**Syntax:**

**#/bin/bash  
for <var> in <value1 value2 ... valuen>  
do  
 <command 1>  
 <command 2>  
 <etc>  
done**

*#/bin/bash*

*#Start of for loop*

*for a in 1 2 3 4 5 6 7 8 9 10  
do*

*# if a is equal to 5 break the loop  
if [ $a == 5 ]  
then  
break  
fi*

*# Print the value  
echo “Iteration no $a”  
done*

**Array**

* **Array is a collection of values.**
* **Array is a list containing multiple values under a single variable entity.**
* **Array values are typically called elements.**
* **In Shell scripting, array can hold element of same type or different types**
* **The array form that references the data elements using indexes starting from ‘0’.**

**EXPLANATION**

* [**echo**](https://linuxsimply.com/echo-command-in-linux/#Example_3_Using_the_echo_Command_to_View_Variables_in_the_Command_Line)**:**Bash command used to set the array variable attribute.
* **${}:**Used to retrieve the value of a variable**.**
* **arr\_name:**The name of the array you want to assign.
* **[@]/[\*]:**Points out the referencing of all elements of the array
* **[index\_num]:** Number used to locate the position of the intended array item.
* **${#arr[@]} :** # sign give you the length of array
* **unset[@] or unset[3]:** To delete array or particular element in array use unset function.

**looping over arrays**

* We can use**for loops**to access the bash array elements using the following syntax:

for items in ${arr\_name[@]}

do

echo $items

done

**EXPLANATION:-**

* **for:**Keyword to use for loop for iterating the array items.
* **items:**Example of a user-defined variable holding the current array element.
* **in:**Indicates the set of array values to iterate over.
* **${arr\_name}:**The array expansion to get the element values.
* **[@]:**Indicates that you want to iterate over an array’s elements.
* **Array Indexing**: ${**!**array[@]} retrieves array indices

**How to get the status of last command executed?**

* **echo $?** prints the exit status of the last command.
* Use this to determine if the last command was successful (**exit status 0**) or if an error occurred **(Note: exit status will be any different number).**

**Regex:**

* **Regex** or Regular Expression is a string pattern that represents nore than one string or several sequence of characters.
* Regular Expressions are made up of two types of characters
* They are:
* Ordinary Characters or Literals-> such as space, underscore(\_),A-Z,a-z,0-9
* Special Characters or Meta Characters -> |, ^ , $ , . , ? , + , \* , () and [] and \
* They are two types of regular expressions
* They are
* BRE – Basic Regular Expressions (vi, sed, grep)
* ERE – Extended Regular Expressions (awk)
* We can use extended regex in grep and sed just by passing -E option.