**Introduction to Eval in Shell Script**

Eval command is like any other command responsible for getting the popularity shell script is enjoying today. This is one of the most powerful commands and besides being powerful, it provides flexibility as well. This command enables us in a combination of all the arguments together into a combined expression and then executes that. At the end, the execution returns an exit status of the command. This though of execution can be write related to the execution of a script. We combine all the command and as soon as we execute the script, we execute the command in the same chronological order. In quite a number of instances, we have seen people using the concept of exec and eval interchangeably. But the usage is not correct at all. Both are completely different genres of beasts. In the case of exec, the execution of the command doesn’t fork and just replaces the shell. In layman terms, it simply means that the process id of exec inside a shell script is the same as the process id of the shell script itself. But in case of eval, a child process is created, and the commands are executed there and not in the process id of the shell itself. Hence you would see a different process id of the eval command itself.

Another important concept is that eval is a part of POSIXwhich is nothing, but a standard maintained by IEEE for keeping compatibility between operating systems. The reason for being a benchmark or being a standard system is simply because of Unix being neutral from a manufacturer’s perspective.

**Syntax of Eval in Shell Script**

Now let us come to the syntax part of eval command. The syntax goes on as:

eval [argument]

Although the syntax looks pretty plain, but the power lies in the simplicity itself. If we want to get deeper on what exactly is performed. The argument is converted like a single string and that acts as an input to another child process of the shell where the resulting commands are executed.

**Process of Eval**

Let us see a process of eval in action.

If the command is as follows:

eval var\_1='$'$var\_2

what it will eventually do is take the var\_1=’$’$var\_2 as a string itself and then try to execute that in another child process. Now in another process, the $var\_2 will be executed first and the corresponding value will be an output, which can be a string and along with an escape character, var\_1 will attain the value stored in the variable $var\_2 output, which would be a string.

The eval statements find a lot of use cases in times when we want to access the index variable of an argument. One can also type eval –help in the command line to get quick help on the eval documentation.

**Examples to Implement of Eval in Shell Script**

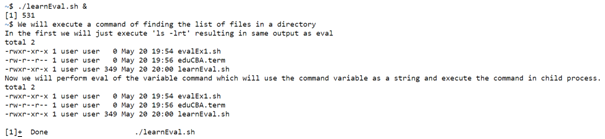
In this section let us look at some hands-on some examples from the real world in order to understand the eval function in detail. Here we would start with the basics of eval function, then move on to an example where we would print value of a variable assigned to another variable, and in the final one, we would look at a simulation of accessing the index use case we talked about earlier.

**Example #1**

**Code:**

command="ls -lrt"  
echo "We will execute a command of finding the list of files in a directory"  
echo "In the first we will just execute '$command' resulting in same output as eval"  
$command  
echo "Now we will perform eval of the variable command which will use the command variable as a string and execute the command in child process."  
eval $command

**Output:**

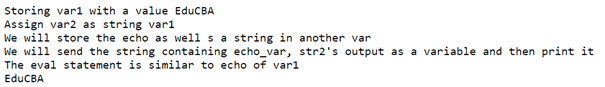


**Example #2**

**Code:**

#!/bin/bash  
echo "Storing var1 with a value EduCBA"  
var1="EduCBA"  
echo "Assign var2 as string var1 "  
var2=var1  
echo "We will store the echo as well s a string in another var"  
echo\_var="echo"  
echo "We will send the string containing echo\_var, str2's output as a variable and then print it"  
echo "The eval statement is similar to echo of var1"  
eval $echo\_var \${$var2}

**Output:**

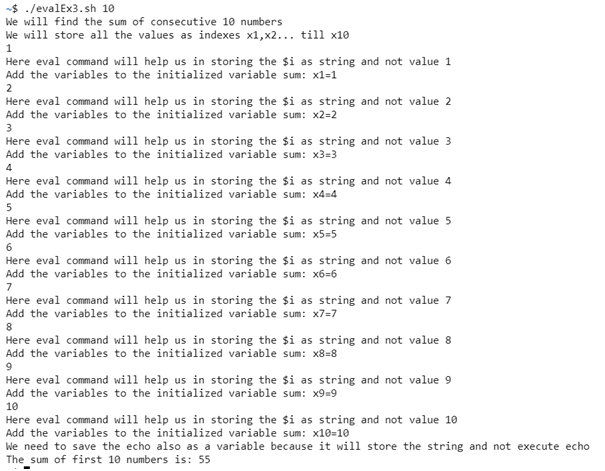


**Example #3**

**Code:**

#!/bin/bash  
echo "We will find the sum of consecutive $1 numbers"  
addition=0  
echo "We will store all the values as indexes x1,x2... till x$1"  
for ((i=1;i<=$1;i++))  
do  
echo $i  
echo "Here eval command will help us in storing the \$i as string and not value $i"  
evalx$i=$i  
echo "Add the variables to the initialized variable sum: x$i=$i"  
addition=$(($addition+x$i))  
done  
echo "We need to save the echo also as a variable because it will store the string and not execute echo"  
command="echo 'The sum of first $1 numbers is:'"  
eval $command $addition

**Output:**



**Conclusion**

In conclusion, we have simulated an index accessing use case and tried to simplify the process of understanding eval through our 3 examples. If we carefully note that in some cases the eval is the same as normally executing the command, as evident from the first example. From the example 2 onwards, we were able to feel the power of eval and finally in example 3, we are well appreciating the use of eval in shell script and why shell script has attained such popularity.

**Recommended Articles**

This is a guide to Eval in Shell Script. Here we discuss the Introduction to Eval in Shell Script and its Examples along with its Code Implementation. You can also go through our other suggested articles to learn more –