TCL - COMMANDS

As you know, Tcl is Tool command language, commands are the most vital part of the language. Tcl commands are built in-to the language with each having its own predefined function. These commands form the reserved words of the language and cannot be used for other variable namings. The advantage with these Tcl commands is that, you can define your own implementation for any of these commands to replace the original built-in functionality.

Each of the Tcl commands validates the input and it reduces the work of the interpreter.

Tcl command is actually a list of words, with the first word representing the command to be executed. The next words represent the arguments. In order to group words into a single argument, we enclose multiple words with "" or {}.

The syntax of Tcl command is as follows.

```
commandName argument1 argument2 ... argumentN
```

Let's see an simple example of Tcl command.

```
#!/usr/bin/tclsh

puts "Hello, world!"
```

When above code is executed, it produces following result.

```
Hello, world!
```

In the above code, puts is the Tcl command and "Hello World" is argument1. As said before, we have used "" to group two words.

Let's see another example of Tcl command with two arguments.

```
#!/usr/bin/tclsh

puts stdout "Hello, world!"
```

When above code is executed, it produces following result.

```
Hello, world!
```

In the above code, puts is the Tcl command,stdout is argument1 and "Hello World" is argument2. Here stdout makes the program to print in the standard output device.

Command substitution

In command substitutions, square brackets are used to evaluate the scripts inside the square brackets. A simple example to add two numbers is shown below.

```
#!/usr/bin/tclsh

puts [expr 1 + 6 + 9]
```

When above code is executed, it produces following result.

```
16
```

Variable substitution

In variable substitutions, $ is used before the variable name and this returns the contents of the variable. A simple example to set a value to a variable and print it is shown below.

```
#!/usr/bin/tclsh

set a 3
puts $a
```

When above code is executed, it produces following result.

```
3
```

Backslash substitution

These are commonly called escape sequences with each backslash followed by letter having its own meaning. A simple example for newline substitution is shown below.

```
#!/usr/bin/tclsh

puts "Hello\nWorld"
```

When above code is executed, it produces following result.

```
Hello
World
```