

Algorhythm**Network Simulator 2**

- [Ns2-Installation on Ubuntu 10.04](#)
- [Ns2-Installation on Ubuntu 14.04](#)
- [How to run Tcl](#)
- [Tcl Basics](#)
- [OTcl Basics](#)
- [Node Command](#)
- [Link Command](#)
- [Agents](#)
- [Application](#)
- [Trace file - Wired](#)
- [Trace file - Wireless\[old-format\]](#)
- [Trace file - Wireless\[new-format\]](#)
- [Mobile networks simulation](#)
- [First wired simulation](#)
- [First wireless simulation](#)
- [Energy Model](#)
- [BlackHole Attack](#)
- [Mesh Topology](#)

Tcl Tutorials

[Pre - Instalation on Ubuntu](#)[Next - Learn OTcl Programming](#)

TCL is an interpreted language. Each instruction is a command in tcl program. TCL programming is used to write simulation script in ns2.

Following are the topics which you need to learn in order to write simulation script.

Topics

- [Variables in TCL](#)
- [Basic Operations](#)
- [If-else](#)
- [Loop](#)
- [Array](#)
- [Procedure](#)
- [File Handling](#)

Variables in TCL

set command is used to make a variable and assign value to them. This command is also used to fetch value of variable.

Syntax :

```
set <variable_name> <variable_value>
```

▸ Ring Topology

▸ Star Topology

▸ LAN Simulation

▸ AODV simulation

▸ DSDV simulation

▸ DSR simulation

AWK Scripts

Thought of the Day

Keep your eyes on
the stars, and your
feet on the ground

Theodore Roosevelt

Above command will create a variable with given name and assign value. If value is not specified then above command will show value stored in variable. Let's take one example

Example :

```
1 set x 10
2 set name "john"
3 set price 12.2
```

First command creating a variable with name x and value 10. In second command string "john" is assigned to variable name. Third command assigning float value to variable price.

IMPORTANT !

if value is not specified in set command then it will return value of variable.

Exmample :

```
set x ;# this command will return value of x
```

For deleting variable unset command is used.

Unset command :

```
unset x
```

Print/Use variable's value

Value of any variable can be used by using \$ symbol. Let's say we want to assign value of x variable to another variable y.

Command :

```
set y $x
```

Above command first replace \$x by it's value which is 10 then 10 will be assigned to variable y.

For printing purpose put command is used.

put command :

```
1 | puts "hello world"
2 | # Output: helloworld
```

Above command will print hello world message on the screen and take cursor to new line. If we want to print next message on the same line then *-nonewline* option has to be specified. # symbol is used to provide comments.

Program :

```
1 | puts "hello world"
2 | puts -nonewline "welcome"
3 | puts " to TCL"
4 |
5 | # Output:
6 | # hello world
7 | # Welcome to TCL
```

In above code we have three commands. First command will print hello world on screen and then move cursor to next line. Second command will print welcome on screen only. Third command will print to TCL message at the end of second line.

Program :

```
1 | set x 10
2 | set y 20
3 | puts "value of x is $x"
4 | puts "value of y is $y"
5 |
6 | # Output:
7 | # value of x is 10
8 | # value of y is 20
```

Performing basic operations

In tcl expr command used to perform operations. expr command takes entire expression as arguments. Let's say we want to add

two variable x and y

Program :

```
1 set x 10
2 set y 20
3 expr $x + $y
4
5 #Output:
6 # 30
```

expr command in above code add both variable and print their value.

In above code to assign result of addition to third variable z, brackets are used. Command written in bracket replaced by it's result. For assigning result to variable z set command has to be used.

So command will be like

Syntax :

```
set z <result must be here>
```

For placing result of addition in set command we use brackets.

Program :

```
1 set x 10
2 set y 20
3 set z [expr $x + $y]
4 puts "result of addition: $z"
5
6 # Output:
7 # result of addition: 30
```

Let's take one more example for use of expr command to calculate area of circle

Example :

```
1 set radius 7
2 set pie [expr 22.0/7]
3 set area [expr $pie*$radius*$radius]
4 puts "Area of circle: $area"
5
6 # Output:
7 # Area of circle: 154.0
```

If-else

TCL also has construct for if-else. Syntax for if-else is given below.

Syntax :

```
if {condition} {  
  if-body  
}  
  
# Another syntax  
if [ expr condition ] {  
  if-body  
}
```

Let's take one example for checking whether number is less than 10 or not.

Program :

```
1  set x 10  
2  if { $x < 50 } {  
3    puts "x is less than 50"  
4  }  
5  
6  
7  if [ expr $x == 20 ] {  
8    puts "x is equal to 20"  
9  } else {  
10   puts " x is not equal to 20"  
11 }  
12 # Output:  
13 x is less than 50  
14 x is not equal to 20
```

IMPORTANT !

Opening brace for if-body must be in same line of if-condition. If you place opening brace at new line it will create error.

Program :

```
1  # Program to Calculate division of student.
```

```
2  set english 40
3  set maths 60
4  set science 60
5  set total [ expr $maths+$english+$science ]
6  set per [expr $total/3]
7  if { $per >60 } {
8  puts "first Division "
9  } elseif { $per >50} {
10 puts "Second division "
11 } elseif { $per > 33} {
12 puts "third division "
13 } else {
14 puts "Failed "
15 }
16 #Output:
17 # Second division
```

Loop

In TCL while loop has similar syntax as in c. Only different is in condition specification.

While Loop

Syntax :

```
while { condition } {
    while-body
}
```

IMPORTANT !

Note:Opening brace for while must be in same line of while-condition. If you place opening brace at new line it will create error.

Example of printing event values form 0 to 100 :

```
1  set s 0
2  while { $s < 100 } {
3  puts $s
4  set s [expr $s+2]
5  }
```

For loop

For loop in tcl also has three parts initialisation, condition and increment or decrement.

Syntax :

```
for { initialisation} { condition} {increment/decrement} {  
loop-body  
}
```

Example:

```
1  for {set i 0} { $i < 10} { incr i} {  
2  puts $i  
3  }  
4  # Output:  
5  #0  
6  #1  
7  #2  
8  #3  
9  #4  
10 #5  
11 #6  
12 #7  
13 #8  
14 #9
```

In the above program first part of for loop setting value of i as 0. second part has a condition that value of i must be less than 10. Third part increasing value of i. incr is special instruction of tcl which increase i by 1.

Array

Array is a collection of homogeneous data. TCL also support named array in which string is given as named variable.

Example :

```
1  set a(0) 10  
2  set a(1) 30  
3  puts $a(0)
```

```
4 #####Another Way
5 array set b {10,30}
6 puts $b(0)
7
8 # Output:
9 10
```

Procedure

Multiple commands can be combined to make a new command. This can be done by making a procedure. Procedure is similar to function in 'c'.

Syntax:

```
proc <procedure name> { } {
  procedure-body
}
```

Let's take one example of factorial to understand this concept.

Program :

```
1 # Program to make procedure for factorial
2 proc fact { a } {
3   set fact 1
4   for {set i 1} {$i <= $a} {incr i} {
5     set fact [expr $fact * $i]
6   }
7   puts $fact
8 }
9 # Calling procedure
10 fact 5
11
12 #Output:
13 # 120
```

In above program we have created one fact procedure with one argument. In this procedure we have used loop to calculate factorial. For calling a procedure procedure name and arguments need to be specified.

Global Variable in Procedure

In procedure all variable known as local. If you wants to access some variable which are created outside of procedure then global command is used.

Program :

```
1  set x 20
2  proc demo { } {
3    global x
4    set x 30
5  }
6  demo
7  puts $x
8  #Output:
9  #20
```

In procedure we have specified that x is not a local variable. So now it will access variable x which is created outside of procedure.

File Handling

Before performing any operation on file, that file need to be opened. In tcl open command is used to open a channel with file to perform read/write operation. File can be opened in various mode which allow which operation is allowed. These modes are

Read only Mode (r)

Write Only Mode(w)

Append Mode (a)

Read and Write (r+)

Write and Read (w+)

Program :

```
1  # For opening file
2  # open <file_name> <acc_type>
3
4  open one.txt r
5
6  # For assigning open file handle to variable.
7  set f_in [open one.txt r]
```

Above code will open "one.txt" file in read mode. It means only read operation can be performed on this file. Last command in above code open "one.txt" in read mode and then assign it to variable `f_in`, which can be used further for file processing.

Writing File

`puts` command is used for writing data in file.

Syntax :

```
1 # Syntax for writing in file.
2 # puts <file_handle> <data|variable_value>
3
4 set f [open one.txt w]
5 puts $f "welcome to tcl"
6
7 close $f
```

Reading File

`gets` command is used for reading data from file.

Syntax :

```
1 # Syntax for reading from file.
2 # gets <file_handle> <variable_value>
3
4 set f [open one.txt r]
5 gets $f data
6
7 puts $data
8 close $f
```

[Pre - Installation on Ubuntu](#)

[Next - Learn OTcl Programming](#)

4 Comments

Sort by **Oldest**

Add a comment...

**Ritesh Kumar** ·

NIT Patna

nice user friendly

Like · Reply · Feb 6, 2017 10:44pm

**Saurav Bilung** ·

Manit Bhopal

Nice tutorial series man

Like · Reply · 1 · Feb 27, 2017 5:04pm

**Gethzi Akila** ·

Junior Research Fellow at National Engineering College

very nice tutorial

Like · Reply · Apr 29, 2017 12:07pm

**Ahsan Habib Tareq** ·

Branch Coordinator at IEEE Student Branch DU

Nice tutorials on NS2 basics.

Search

JAVA

Java-Installation

Java Execution

NS2

Ns2 Installation

Ns2 Commands

JAVA SWING

Create Frame

Event Handling

NETWORKING

Socket Basics

Simple Client-Server

Java-features	Ns2 Examples	Layout Managers	Port Scanner
Platform-independent	OTcl tutorials	Mouse Event	Remote Method
javadoc command	TCL tutorials	Develop Calculator	Invocation
			Chat server development

© 2015 - 2016 [jgyan.com](http://www.jgyan.com). All rights reserved.

