OTcl: The User Language

As mentioned in the overview section, NS is basically an OTcl interpreter with network simulation object libraries. It is very useful to know how to program in OTcl to use NS. This section shows an example Tcl and OTcl script, from which one can get the basic idea of programming in OTcl. These examples are from the 5th VINT/NS Simulation Tutorial/Workshop. This section and the sections after assumes that the reader installed NS, and is familiar with C and C++.

Example 1 is a general Tcl script that shows how to create a procedure and call it, how to assign values to variables, and how to make a loop. Knowing that OTcl is Object-orieneted extension of Tcl, it is obvious that all Tcl commands work on OTcl - the relationship between Tcl and Otcl is just same as C and C++. To run this script you should download ex-tcl.tcl, and type "ns ex-tcl.tcl" at your shell prompt - the command "ns" starts the NS (an OTcl interpreter). You will also get the same results if you type "tcl ex-tcl.tcl", if tcl8.0 is installed in your machine.

```
# Writing a procedure called "test"
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
        if {$k < 5} {
            puts "k < 5, pow = [expr pow($d, $k)]"
        } else {
            puts "k >= 5, mod = [expr $d % $k]"
        }
    }
}

# Calling the "test" procedure created above
test
```

Example 1. A Sample Tcl Script

In Tcl, the keyword **proc** is used to define a procedure, followed by an procedure name and arguments in curly brackets. The keyword **set** is used to assign a value to a variable. [**expr** ...] is to make the interpreter calculate the value of expression within the bracket after the keyword. One thing to note is that to get the value assigned to a variable, **$** is used with the variable name. The keyword **puts** prints out the following string within double quotation marks. The following shows the result of Example 1.

```
k < 5, pow = 1.0
k < 5, pow = 1120.0
k < 5, pow = 1254400.0
k < 5, pow = 1404928000.0
k < 5, pow = 1573519360000.0
k >= 5, mod = 0
k >= 5, mod = 4
k >= 5, mod = 0
k >= 5, mod = 0
k >= 5, mod = 4
```

The next example is an object-oriented programming example in OTcl. This example is very simple, but shows the way which an object is created and used in OTcl. As an ordinary NS user, the chances that you will write your own object might be rare. However, since all of the NS objects that you will use in a NS simulation programming, whether or not they are written in C++ and made available to OTcl via the linkage or written only in OTcl, are essentially OTcl objects, understanding OTcl object is helpful.

```
# add a member function call "greet"
Class mom
mom instproc greet {} {
    $self instvar age_
    puts "$age_ year old mom say:
    How are you doing?"
}

# Create a child class of "mom" called "kid"
# and overide the member function "greet"
Class kid -superclass mom
kid instproc greet {} {
    $self instvar age_
    puts "$age_ year old kid say:
    What's up, dude?"
}

# Create a mom and a kid object, set each age
set a [new mom]
$a set age_ 45
set b [new kid]
$b set age_ 15

# Calling member function "greet" of each object
$a greet
$b greet
```

Example 2. A Sample OTcl Script

Example 2 is an OTcl script that defines two object classes, "mom" and "kid", where "kid" is the child class of "mom", and a member function called "greet" for each class. After the class definitions, each object instance is declared, the "age" variable of each instance is set to 45 (for mom) and 15 (for kid), and the "greet" member function of each object instance is called. The keyword **Class** is to create an object class and **instproc** is to define a member function to an object class. Class inheritance is specified using the keyword **-superclass**. In defining member functions, **$self** acts same as the "this" pointer in C++, and **instvar** checks if the following variable name is already declared in its class or in its superclass. If the variable name given is already declared,

the variable is referenced, if not a new one is declared. Finally, to create an object instance, the keyword **new** is used as shown in the example. Downloading ex-otcl.tcl and executing "ns ex-otcl.tcl" will give you the following result:

```
45 year old mom say:
    How are you doing?
15 year old kid say:
    What's up, dude?
```