Tcl Debugging Tools

Version 0.1.0

Alex Baker

https://github.com/ambaker1/dbug

March 6, 2023

Abstract

Since OpenSees is a script-based finite element analysis software, it can be difficult to debug a model or analysis when problems arise. Typically, *puts* statements are the extent of debugging a script written in Tcl, but this method can be cumbersome for more complex scripts. Towards making OpenSees Tcl more user-friendly, the "dbug" package makes debugging easy, with tools such as the pause command.

Requires the package wob, which can be downloaded at https://github.com/ambaker1/wob

Pausing a Script

The *pause* command pauses a Tcl script, prints the file and line number, and enters command-line mode, allowing the user to query variables and insert code into an analysis. If the command entered while paused returns an error, the error message will be displayed and the script will remain paused. If the command entered is "return", the pause will be exited and the corresponding result and options will be passed to the caller. For example, a loop can be broken by entering *return -code break* in pause mode. Pressing enter with no commands will simply continue the script.

```
pause
```

Example 1: Pausing an analysis

Code:

```
pause
```

Output:

```
PAUSED...
File: C:/User/Documents/MyFile.tcl
Line: 407
>
```

Note: The pause command cannot be used in interactive mode (no need to pause when in command-line mode), but a pause can be used within a pause.

Advanced Tcl Debugger

The *debug* command parses a Tcl script, printing out intermediate steps of Tcl evaluation. Optionally, a recursive maximum depth to step into procedures can be specified. Commands are shown in the command window with a > prefix, and procedure depth is shown with asterisks. Each command is matched with a result line, thus showing the commands and their intermediate results. Note that this will slow down execution dramatically.

```
debug $script <$depth>
```

$script	Tcl script to debug.
$depth	Optional recursive depth to step into procedures (default 0)

Example 2: Debugging a procedure

Code:

```
proc add {a b} {
    return [expr {$a + $b}]
}
set a 5
set b 7
debug {
    add [expr {$a*2}] $b
} 1
```

Output:

```
------------------------------------------------------------------------------
COMMAND:
> add [expr {$a*2}] $b
------------------------------------------------------------------------------
STEPS:
> expr {$a*2}
10
> add 10 7
****> expr {$a + $b}
****17
****> return 17
17
------------------------------------------------------------------------------
```

Verbose Command Evaluation

The command > simply prints the fully substituted command, with prefix ">", and the results. It is kind of the opposite of the comment symbol "#", and is not intended to be used in interactive mode.

```
> $arg1 $arg2 ...
```

`$arg1 $arg2 ...`	Command arguments.

Example 3: Verbose evaluation

Code:

```
set a 5
set b 7
> expr $a + $b
```

Output:

```
> expr 5 + 7
12
```

Unit Testing

The command *assert* can be used for unit testing of Tcl scripts. Throws error if the statement is false. If the statement is true, it simply returns nothing and the script continues.

```
assert $value1 <$op $value2>
```

`$value1`	Value to test.
`$op`	Comparison operator. Default "==".
`$value2`	Comparison value. Default "true".

Example 4: Validation for unit testing

Code:

```
assert [string is double 5.0]; # Asserts that 5.0 is a number
assert [expr {2 + 2}] == 4; # Asserts that math works
```

Viewing Variables

The *pvar* command is a short-hand function for printing the name and value of Tcl variables, including arrays.

```
pvar $name1 $name2 ...
```

$name1 $name2 ...	Name(s) of variables to print

Example 5: Printing variables to screen

Code:

```
set a 5
set b(1) 2
set b(2) 3
pvar a b
```

Output:

```
a = 5
b(1) = 2
b(2) = 3
```

Interactive Workspace Viewer

Alternatively, the command *viewVars* can be used to pause a Tcl script and opens up an interactive table of all variables in the current scope and their values. Variables cannot be edited in the variable viewer window, but their values can be selected and copied. This command in particular requires the packages "Tk" and "Tktable".

```
viewVars
```