A Debugger for Tcl Applications

Don Libes

National Institute of Standards and Technology

Overview

- Motivation
- Look & Feel
- Design & Implementation
- Problems Encountered
- Performance
- Future

Motivation

- Sufficiently long scripts are painful to develop
  - On some days, “sufficiently long” means: **`` `wc -l` > 2 ``**

Good news/Bad news

- Tcl trap facility (Tcl_CreateTrace)
  - It's perfect . . . enough
- Re-use all Tcl and applications commands
  - Example debugger interaction:

```
debug> set fruits {apple banana mango}
apple banana mango
debug> set fruits
apple banana mango
debug> llength $fruits
3
debug>
```

Commands Are Simple

- Step (step into); Next (step over)

```
n
s
s 2
s $var
s [foo "banana" [expr 7 + [bar 2]]]
```

- Return to caller

```
r
```

- Continue

```
c
```

Breakpoints – By Location

- Line numbers

```
b 24
b init.tcl:24
```

- Patterns

```
b -re "foo"
b -re "^foo "
b -re "^foo$"
b -re "^foo"
b -re { $foo}
b -re {\[foo]}
```

Breakpoints – By Expression

```
b -re "^foo " if {$bar == 17}
b -re "^foo " if {[info exists bar]}
b -re "^foo " if {...
        . . .
}
b -re "^foo (.*)" if {17 == [bar $dbg(1)]}

b if {$bar == 17}

b {$bar == 17}

b $bar==17
```

Breakpoints – Actions

- **Any actions that can be invoked from the debugger**

```
b -re "^foo " if {$bar > 0} then gorp

b -re "^foo " if {$bar > 0} then {
     puts "bar = $bar"
}

b -re "^foo " then {
     puts "bar = $bar"
}
```

Debugger Actions

- Only a single debugger command at a time

```
b -re "^foo " then {
     puts "bar = $bar"
     c
}

b -re "^return" then s
```

Scope Commands

- References to other scopes

```
uplevel {set foo}
global foo
```

- Move scope up, down, to

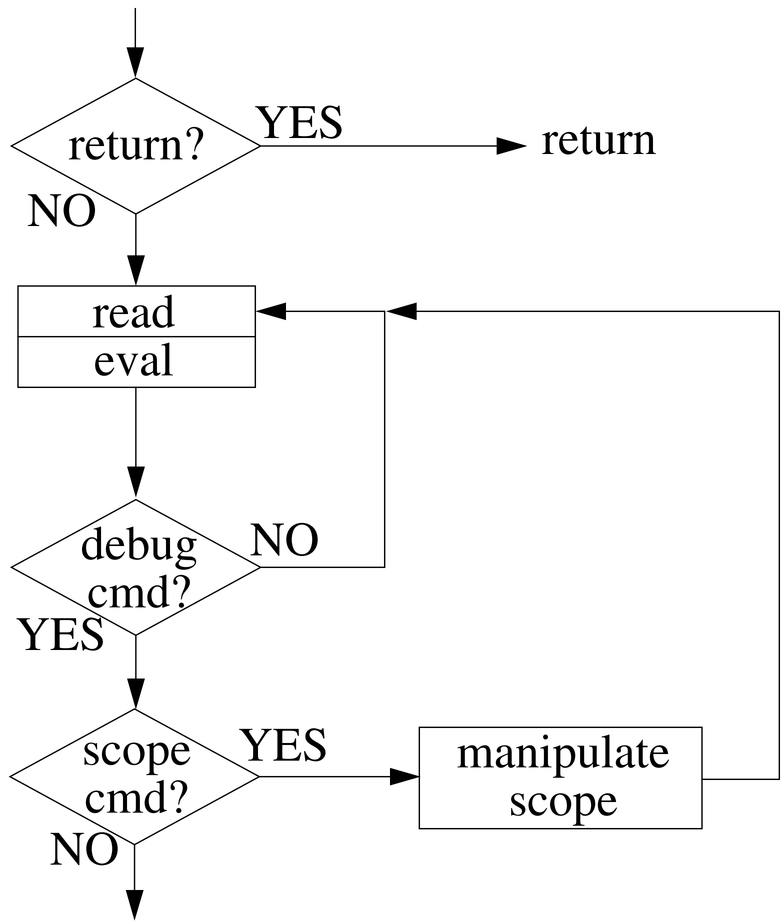
```
u
d 3
d #4
```

- Where am I?

```
w
```

Implementation

- Trace Facility
  - debugger_trap() called before every command



Problems

- Scope Stack Access
- Line Numbers & Filenames
- Command & Argument Representation

Impact

- Space: 1-2% (13K)
- Time: 1-10%
- Money: 0%

Future

- Line numbers?
- Tcl_Interp?
- Tk’s send?

Summary

- Debugger Philosophy
  - As simple as possible
  - Leverage – existing commands
  - To stay out of the way
- Starting Point For Much More Sophisticated Debuggers?