Coding Techniques for Reducing Code Maintenance

Clifton Flynt
Flynt Consulting Services

Coding Techniques for Reducing Code Maintenance

Clifton Flynt

Flynt Consulting Services

Abstract

Tcl/Tk supports code constructs that can reduce the amount of code that needs to be changed when code is modified.

The following examples and brief discussion explain some code conventions that can reduce code maintenance.

Shell or C programmers are familiar with using a switch statement to parse a command line. In Tcl the command line parsing code can reformat the arguments into Tcl commands. This allows new command line arguments to be added without code modification.

One such convention is "-varName value", which can be parsed by a set of code resembling example 1.

The names of global variables can be placed in a list to be evaluated by procs which need access to them. This creates a single point of change when new variables need to be declared global. Example 2 shows sample code.

Menu construction can be data driven instead of code driven. For example, a program which processes the contents of files can have the file selection menu built with the Tcl `glob` command. This allows new files to be automatically included in the menu. The traditional method of hardcoding a list of items to place in a menu would require a code modification whenever new files are added (or a new BaseDirectory is selected from the command line). See Example 3.

Saving and restoring state arrays can also be be data driven instead of code driven. Tcl can report all of the indices of an array. This list can then be used to drive the code which saves these values. By saving the variables as a Tcl command string, the state can be restored with the source command. See Example 4.

Further examples and discussion are included in the posters.

These conventions were developed while writing, extending and maintaining TclTutor.tk which is available at:

`http://www.msen.com/~clif.`

Clif Flynt can be reached as *clif@clif.ypsi.mi.us*

Example 1.

```
# Scan command line for -Varname Value combinations

for {set i 0} {$i < [llength $argv]} {incr i} {
  set arg [lindex $argv $i]
  if {[string first "-" $arg] == 0} {
    incr i;
    eval [list set [string range $arg 1 end] [lindex $argv $i]]
    } else {
    puts "Bad argument: $arg"
    }
  }
```

Example 2.

```
# Global variables are all declared in a single location

set globalList [list errorCode errorInfo stateArray argv argc]

proc GenericProc {} {
  global $globalList; eval "global $globalList";
  ...
  }
```

Example 3.

```
# Add a list of files to a menu

foreach file [glob $stateArray(BaseDirectory)] {
  $menu add command -label $file -command "ProcessFile $file"
  }
```

Example 4.

```
# Save state as a source-able file.

puts $stateFile "array set stateArray [array get stateArray]"
```