

TCL and Expect

William Tracy

Thursday, February 19, 2009

What is TCL?

- ▶ Unix scripting language
- ▶ Intended for embedding in applications
- ▶ Shell-like syntax
- ▶ Low memory footprint

Control flow

Listing 2: File 1-if

```
1 #!/usr/bin/tclsh
2
3 set flag true
4 if {$flag} {
5     puts True!
6 } else {
7     puts False!
8 }
```

Loops

Listing 3: File 2-loops

```
1 #!/usr/bin/tclsh
2
3 set input n
4 while {$input != y} {
5     puts "Would you like to quit? <y/n>"
6     gets stdin input
7 }
8
9 for {set i 0} {$i < 10} {incr i} {
10     puts $i
11 }
```

Example

Listing 4: File install

```
1  #!/usr/bin/tclsh
2
3  set fail true
4
5  puts "Install _optional _components? <y/n>"
6  gets stdin data
7  if {$data != y} {
8      exit
9  }
10
11 if {$fail} {
12     puts "Hardware _not _supported. _Install _will _cause _spontaneous"
13     puts "combustion. _Continue? <y/n>"
14     gets stdin data
15     if {$data != y} {
16         exit
17     }
18 }
19
20 puts "Install _will _take _10Gb. _Continue? <y/n>"
21 gets stdin data
```

Whitespace

Listing 5: File 3-ifbroken

```
1 #!/usr/bin/tclsh
2
3 set flag true
4 if {$flag}
5 {
6     puts Yay!
7 }
```

Whitespace

Listing 6: File 4-iffixed

```
1 #!/usr/bin/tclsh
2
3 set flag true
4 if {$flag} {
5     puts Yay!
6 }
7 if {$flag} \
8 {
9     puts Yay!
10 }
```

Whitespace

Listing 7: File 5-comments

```
1 #!/usr/bin/tclsh  
2  
3 puts " Blabla" # Witty comment  
4  
5 puts " Blabla"; # Witty comment
```

Grouping

- ▶ Quotes group arguments
- ▶ Braces disable substitution in a group
- ▶ Square brackets are replaced by the execution of that command

Listing 8: File 6-brackets

```
puts [expr 2 + 2]
```

Data Structures

Listing 9: File 7-structures

```
1 #!/usr/bin/tclsh
2
3 set mylist {a b c d}
4 puts $mylist
5 puts [lindex $mylist 0]
6 puts [llength $mylist]
7
8 set myarray(foo) bar
9 set myarray(bazz) buzz
10 parray myarray
11 puts $myarray(foo)
```

Procedures

Listing 10: File 8-proc

```
1 #!/usr/bin/tclsh
2
3 proc sum {arg1 arg2} {
4     set x [expr {$arg1 + $arg2}]
5     return $x
6 }
7 puts "The sum of 2+3 is: [sum 2 3]"
```

What is Expect?

- ▶ Extension to TCL
- ▶ Spawns and manages child processes
- ▶ Handles stdin and stdout of children

Spawn

Listing 11: File 9-spawn

```
1 #!/usr/bin/expect  
2 spawn cat 9-spawn  
3 puts $spawn_id
```

Expect

Listing 12: File 10-expect

```
1 #!/usr/bin/expect
2 spawn cat 10-expect
3 expect expect
4 puts $expect_out(buffer)
5 expect $spawn_id eof
```

Send

Listing 13: File 11-send

```
1 #!/usr/bin/expect
2
3 spawn banner
4 expect Message:
5 send :-) \n
6 expect eof
```

Autoinstall

Listing 14: autoinstall

```
1  #!/usr/bin/expect
2
3  spawn ./install
4
5  expect " Install_optional_components?<y/n>"
6  send y\n
7
8  expect " combustion.Continue?<y/n>" {
9      send "n\n"
10 } " Install_will_take_10Gb.Continue?<y/n>" {
11     send y\n
12 }
13
14 expect eof
```

Introduction to TCL
TCL Fundamentals
TCL oddities
More TCL syntax
Introduction to Expect
Expect fundamentals
Expect examples
The end

Hunt the Wumpus

Introduction to TCL
TCL Fundamentals
TCL oddities
More TCL syntax
Introduction to Expect
Expect fundamentals
Expect examples
The end

The book

