

CSI605

Tcl/Tk

Tcl/Tk

- **Tcl/Tk = Tool Command Language Tool Kit**
- **What is Tcl/Tk**
 - **A shell scripting language**
 - **A multi-platform language**
 - **Extensible interpreter**
 - **Embedded interpreter**
 - **A graphics programming language**
- **Uses of Tcl/Tk**
 - **Rapid prototyping**
 - **Shippable products**
 - **GUI-based products**
 - **Multi-platform products**
 - **Adding GUIs to existing text oriented programs**
 - **Adding new functionality to old code libraries**

Online Tcl/Tk Resources

- **www.tclconsortium.org**
 - **Tcl/Tk Consortium**
- **www.scriptics.com**
 - **Scriptics Home Page**
- **www.neosoft.com**
 - **Tcl/Tk Archives: the official repository for released Tcl/Tk packages**
- **www.net-quest.com/~ivler/cgibook/refs/index.shtml**
 - **Resources: URLs for mailing lists, news groups, books on Perl, Tcl/Tk, HTML and more.**
- **cuiwww.unige.ch/eao/www/TclTk.html**
 - **The WWW virtual library, Tcl and Tk: URLs for tutorials, FAQs, online man pages, books, extensions, applications, and other goodies.**

Invoking the Tcl Interpreter

- **Under LINUX we invoke the TCL interpreter by typing `tclsh` or `wish`**
- **`tclsh` is a text oriented interpreter while `wish` is a text interpreter along with a windows environment**
- **These might be found in `/usr/bin` for example**
- **One can find out where they are by typing `which tclsh` or `which wish`**
- **These can be used to execute a sequence of commands or else interactively**
- **We get out of an interactive session by typing `exit`**

Interactive Session With tclsh or wish

- **Here is our first tcl session**

% tclsh

% puts "Hello, world"

- **Here is our first Tcl program that produces a window**

% wish

% label .1 -text "Hello World "

% pack .1

- **This results in a little window with the words Hello World in it**

A Slightly More Sophisticated Program

- **Here is the Code (example1.tcl)**
label .1 -text "Arguments are \$argv"
pack .1
- **Invoking the code**
wish example1.tcl one two three
- **Resultant Display**



Tcl Syntax

- **Position based**
- **First word of a Tcl command line must be a Tcl command name or a subroutine name**
- **Following words may be subcommands, options, or arguments to the command**
- **The command is terminated by a new-line or semicolon**
- **Valid Commands**
 - `puts "This is cool";`
 - `puts one; puts two;`
- **Invalid commands**
 - `x=4`
 - `puts one put two`
- **Grouping Words**
 - **Spaces between words are important because Tcl uses this to determine which words are commands**
 - **Multiple words that need to be treated as a single word must be included in “ “ or curly braces { }**

Some Examples of the Implication of Spaces

- **Valid Commands**

```
if { $x > 2      } {set greater true}  
set x "a b"  
set x {a b}
```

- **Invalid Commands**

```
if{ $x >2 }      {set greater true}  
if { $x >2 }{set greater true}  
set x a b
```

Comments

- **Valid Comments**

This is a comment

puts "Hello World";#This is an ok comment

- **Invalid Comments**

puts "Hello World" #This one does not work

Data Representation

- Variables don't need to be declared before being used
- Some Simple Variable Examples

set pi 3.14

puts "pi is \$pi"

_____ outputs the string "pi is 3.14"

set y 5

set "weird variable" "Don't Do This Stuff"

Commands Results

```
% set x "apple"
```

```
apple
```

```
% set y [set x "pear"]
```

```
pear
```

```
% put $y
```

```
pear
```

Strings

- All Tcl Data are Strings
- Strings can be considered as lists of words
- Valid String
 - set letters "abcdefg"
 - set msg {Bad input: "Bogus". Try again.}
 - set number 1.776
 - set scientificnotation 10e2
- Bad Strings
 - set badstring "abc
 - set msg "This does not work "Wow" "
 - set mystring jeff solka

Associative Arrays

- **This is an array that uses a string to index the array instead of a number**
- **Examples**
 - set price(bannana) .25**
 - set price(peach) .30**
 - set quantity(bannana) 15**
 - set x(14) 1.25**

Handles

- **These are used to refer to special-purpose objects**
- **Types of Handles**
 - **channel**
 - **A handle that references an I/O device such as a file, serial port or TCP socket**
 - **Returned by an open or socket command**
 - **References by puts, read, close, open, or flush**
 - **graphic**
 - **A handle that refers to a graphics object created by a wish command**
 - **http**
 - **A handle that references data returned by an http:geturl operation. It can be used to access the data returned from the http:geturl command.**

Assigning Values to Variables

- **Command Form**
set varName ?Value?
- **Examples**
 % set y 2
 2
 % set y
 2

Math Operations

- **Increment Operator**
 - **Command Form**
incr varName ?incrValue?

- **Calculation Operator**
 - **Command Form**
expr mathexpression

Simple Operators (In Order of Precedence)

- + ~ !	Unary minus, unary plus, bitwise NOT, logical NOT
* / %	
+ -	
<< >>	Left shift and right shift
<> <= >=	
== !=	
&	Bitwise and
^	Bitwise exclusive or
 	Bitwise or
&&	
 	
x?y:z	If-then-else as in C

Trigonometric Functions

cos(radians)

acos(float)

cosh(radian)

sin(radians)

asin(float)

sinh(radian)

tan(radian)

atan(float)

tanh(float)

hypot(float) is Euclidean distance

atan2(float, float)

Exponential Functions

log(float)

log10(float)

sqrt(float)

exp(float)

pow(float, float)

Conversion Functions

floor(float) **largest integer less than a float**

ceil(float) **smallest integer greater than a float**

fmod(float, float) **remainder**

round(num) **closest int**

double(num)

int(num)

abs(num)

srand(num)

rand()

Examples of Arithmetic Expressions

```
% expr cos(3.14159)  
-0.9999999999996
```

```
% expr sin(3.14159/2)  
0.9999999999999
```

```
% expr floor(exp(1))  
2.0
```

```
% expr pow(2,3)  
8.0
```

Conditionals

- **General Form**

```
if {testexpression1} {  
    body1  
    } ?elseif {testexpression2}{  
        body2  
        } ...?  
    ?else {bodyN}?
```

- **Example**

```
set x 2  
set y 3  
if {$x < $y} {  
    puts "x is less than y";  
}
```

Switch Statements

- General Form

switch ?opt? str pat1 bod1 ?pat2 bod2? ?...? ?defaults

- Example

set x 2

switch -glob \$x {

"1" {puts "one" }

"2" {puts "two" }

"3" {puts "three" }

{[4-8] } {puts greater than 3" }

default{puts "Not 1, 2, or 3" }

}

for loops

- **General Form**

for start test next body

- **Example**

```
for {set i 0} {$i < 3} {incr i} {  
  puts "I is: $i"  
}
```

while loops

- **General Form**
while test body

- **Example**

```
set x 0;  
while { $x < 5 } {  
    set x [expr $x+$x+1]  
    puts "X: $x"  
}
```

String Processing Commands

- **General Forms**

string match pattern string

string tolower string

string first substr string _____ Returns location of the
first instance of a
pattern in a
test string or -1
otherwise.

string length string

format

- **Emulates the behavior of the C sprintf function**
- **Some format definitions**
 - **c** **Arg-in should be decimal integer. Replace the field with the ASCII character value of this integer**
 - **d or I** **Arg-in should be a decimal integer. Replace the field with the decimal representation of this integer**
 - **E or e** **Arg-in should be a numeric value. Replace the field with the scientific notation representation of this integer**
 - **f** **Arg-in should be a numeric value. Replace the field with the decimal fraction representation**
 - **G or g** **Arg-in should be a numeric value. Replace the field with the scientific notation or floating-point representation**
 - **s** **Arg-in should be a string. Replace the field with the argument.**

string Example

```
set dataString  
"field4:Value4:Field1:Value1:FIELD2:VALUE2:  
field3:value3"  
for {set i 1} {$i < 5} {incr i} {  
  
set srch [format "field%d" $i]  
set position($i) [string first $srch  
$dataString]  
if {$position($i) != -1} {  
puts "$srch is found in dataString"  
  
} else {  
puts "$srch is NOT found in  
dataString"  
}  
  
}
```

list Processing Commands

- **list element1 ?element2?...**
 - **Makes a list of the arguments**
- **lappend listname ?element1 element2 ...**
 - **Appends the arguments onto a list**
- **split data ?splitchar?**
 - **Split data into a list and the optional split point**
- **llength list**
- **lindex list index**
 - **Returns a list element referenced by its position in the list**

Array Processing Commands

- **array names arrayName ?pattern?**
 - **returns a list of indices for a given array name**

- **array get arrayName**
 - **returns a list of array indices and the associated values**

array Example

```
set fruit(bannana.cost) .10  
set fruit(bannana.count) 5  
set fruit(peach.cost) .15  
set fruit(peach.count) 3
```

```
foreach item [array names fruit *cost] {  
  set type [lindex [split $item "."] 0]  
  puts "The cost for $type is $fruit($item)"  
  puts "There are $fruit($type.count) units  
  in stock"  
  puts "The $type inventory is worth: \  
    [expr $fruit($item) *  
    $fruit($type.count)]"  
}  
  
puts ""  
puts "indices and values of fruit are:\n  
  [array get fruit]"
```

File IO Example

```
set outputFile [ open "mytestfile" "w"]

puts $outputFile "This is my line 1."
puts $outputFile "This is my line 2."
puts $outputFile "This is my line 3."
puts $outputFile "This is my line 4."

close $outputFile

set inputFile [open "mytestfile" "r"]

set numBytes [gets $inputFile string]

puts "Gets returned $numBytes characters in
      the string: $string"

set string2 [read $inputFile]

puts "Read: $string2"
```

Debugging in Tcl/Tk

- **There are numerous text and graphical oriented Tcl/Tk debuggers**
- **debug**
 - **<http://expect.nist.gov>**
 - **text oriented**
- **Tuba**
 - **<http://www.doitnow.com/~iliad/Tcl/tuba/>**
 - **GUI oriented with support for multiple windows**
- **TdChoose/TdDebug**
 - **<http://www.neosoft.com>**
 - **Allows one to attach to an ongoing wish process**

Suggested References

- *Tcl/Tk for Real Programmers*, Clif Flynt, ISBN 0-12-261205-1, Academic Press.
- *Effective Tcl/Tk Programming: Writing Better Programs in Tcl and Tk*, Mark Harrison and Michael J. McLennan, ISBN-0201634740, Addison-Wesley Professional Programming Computing Series.
- *Graphical Applications With Tcl and Tk*, Eric Foster-Johnson, ISBN-1558515690, IDG Books Worldwide.