

TCL/TK

- It is an Interpreted Programming Language
 - * Build on-the-fly Commands, procedures
 - * Platform-independent
 - * Easy to use for building GUIs.

→ It is easy & fast programming.

How to run your Tcl program:-

- command line (state.cse.ogi.edu or DOS)

- * type "tclsh" to launch the console

- Type your program directly on console.

- Use the command "source" (source filename)

- Double click your .tcl file (if associated)

Output on the Console

- Command: puts "Hello, world!"

Hello World

- Command line (state.cse.ogi.edu or DOS)

- Type "tclsh" to launch the console.

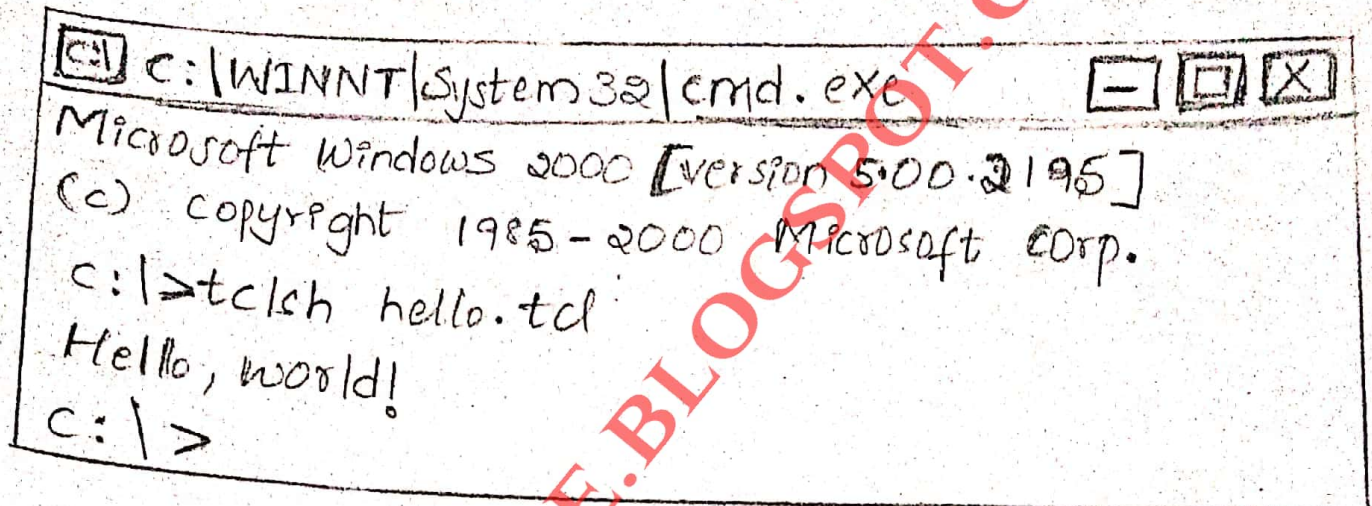
- Type tcl code into console

```
state.cse.ogi.edu - PTTY
state % tclsh
% puts "Hello, world!"
Hello, world!
% exit
state %
```

Hello World

→ Sourced on the console

- Type "tclsh", followed by name of program file #####hello.tcl##### puts "Hello, world!"



```
C:\WINNT\System32\cmd.exe
Microsoft Windows 2000 [version 5.00.2195]
(c) Copyright 1985-2000 Microsoft Corp.
C:\>tclsh hello.tcl
Hello, world!
C:\>
```

Hello world

→ Double clicking your .tcl file (if associated with wish84.exe)

#####hello.tcl#####

Hello.tcl

Wm withdraw

Console show

puts "Hello, world!"

Basic Operations :-

→ Print to Screen (puts)

- puts - newline "Hello, world!"
- puts "!!"

→ Assignment (set)

- set income 32000
- puts "income is \$income"

→ Mathematical Expressions (expr)

- set a 10.0
- expr \$a + 5

Commands :-

→ Unset : destroy a variable - Unset num

→ Info : check whether the named variable has been defined.

```
if { ! [info exists num] }
```

```
{
```

```
    set num 0
```

```
}
```

```
incr num
```

→ Window Commands

- wm withdraw
- console show

Special characters :-

: single-line comments

: in-line comments

\ : escape character & breaking a line

\$: get the value of a variable

- var: name of variable

- \$var: value of variable

[] : evaluates command inside brackets

Control Structures :-

→ If then else

```
set income 32000
```

```
if { $income > 30000 } {
```

```
    puts "$income -- high"
```

```
} elseif { $income > 20000 } {
```

```
    puts "$income -- middle"
```

```
} else {
```

```
    puts "$income -- low"
```

```
}
```

→ while loops

```
set i 0
```

```
while { $i < 100 } {
```

```
    puts "I am at Count $i"
```

```
    incr i
```

```
}
```

for loops

```
for {set i 0} { $i < 100 } {incr i}
```

```
{
```

```
    puts "I am at count $i and going up"
    after 300
```

```
} update
```

```
for {set i 100} { $i > 0 } {set i [expr $i - 1]}
```

```
    puts "I am at count $i and going down"
```

```
}
```

foreach loops

```
set 1stColors {red orange yellow green blue purple}
```

```
foreach c $1stColors
```

```
{
    puts $c
}
```

Procedures :-

→ procedure calls (embedded commands)

- set b [expr \$a + 5]

- puts "The value of b is \$b"

→ Create your own procedure (only value) →

```
proc foo {a b c}
```

```
{
```

```
    return [expr $a * $b - $c]
```

```
}
```

```
puts [expr [foo 2 3 4] + 5]
```

```
proc bar {}
```

```
{
```

```
    puts "I'm in the bar procedure"
```

```
}
```

```
bar
```

Variable Scope

Local and Global variables

set a 5

set b 7

set c 7

proc var_scope { }

{

global a

set a 3

set b 2

set c 1

}

var_scope

puts "The value for a, b, c is: \$a \$b \$c"

Lists in Tcl / Tk :-

→ Everything is a list!

→ Many ways to Create a list

- set mylist [list a b c]
- set mylist "a b c"
- set mylist {a b c} etc...

List Operations :-

- set 1stStudents [list "Fan" "Kristy" "Susan"]
- Puts [lindex \$1stStudents 0]
- Puts [length \$1stStudents]
- lappend \$1stStudents "Peter"
- Puts [linsert \$1stStudent 2 "Tom"]

Lists of lists (of lists...)

- set a [list [list x y z]]
- puts [lindex \$a 0]
- set a [list x [list [list y] [list z]]]

Array Operations

- Associative arrays
- set color (rose) red
- puts [array exists color] (tests if an array with the name "color" exists)
- puts [array exists colour]
- foreach item [array name color]
 - {
 - puts "\$item is \$color(\$item)"
 - }

Regular Expressions:-

- Regsub → Regexp → format

String Operations:-

- set statement "fan is a student"
- puts [string length \$statement]

[File Operations] Miscellaneous Commands

set script

{

set Number1 17

set Number2 25

set Result [expr \$Number1 + \$Number2]

}

eval \$script

Debugging your program:-

- use puts statements.
- Tk-messageBox: pop up a message box
- Tk-messageBox-message "run to here" - type ok
- Tcl pro
- Trace variable, variable Name Operation Procedure