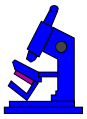
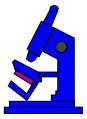



Chapter 5

Tcl/Tk - 1


Environment

- ✔ X - from UNIX/sunfire
- ✔ Cygwin - download ~cs3283/ftp/full.exe (OLD)
- ✔ Native - download www.scriptics.com

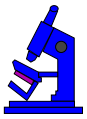


Installing Tcl/Tk

- ✔ If you are using **cygwin-b20**, the wish interpreter is called **cygwish80.exe**.
- ✔ Find **cygwish80.exe**.
- ✔ Copy and call it **wish8.0.exe** for UNIX compatibility.

Installing Tcl/Tk

- ✔ In the first line of your tcl files, you should put **#!/bin/wish8.0**

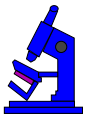


Tcl/Tk demos

- ✔ If you download the file **~cs3283/ftp/demos.tar** and extract it, you will have a series of Tcl/Tk widget examples in **~/Demos**.
- ✔ Change into the directory **~/Demos**, and type **./widget**.

Tcl/Tk tutor

- ✔ There is a Tcl/Tk tutor, and many learn-to-program-Tcl/Tk documents available at many sites on the Internet - if you continue to have trouble, you may wish to try them.

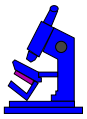


Tcl/Tk

- ✔ Tcl (Tool Command Language) is an interpreted scripting language
- ✔ Pronounced ’tickle’.
- ✔ Tk - X-window toolkit
- ✔ Wish - the windowing shell

Scripting language

- ✔ Difficult to define.
- ✔ Job control languages ...
- ✔ More powerful basic operations
- ✔ Regular-expression pattern matching



Scripting

- ✔ Normally interpreted
- ✔ One line equivalent to 100 lines of C.
- ✔ Overhead

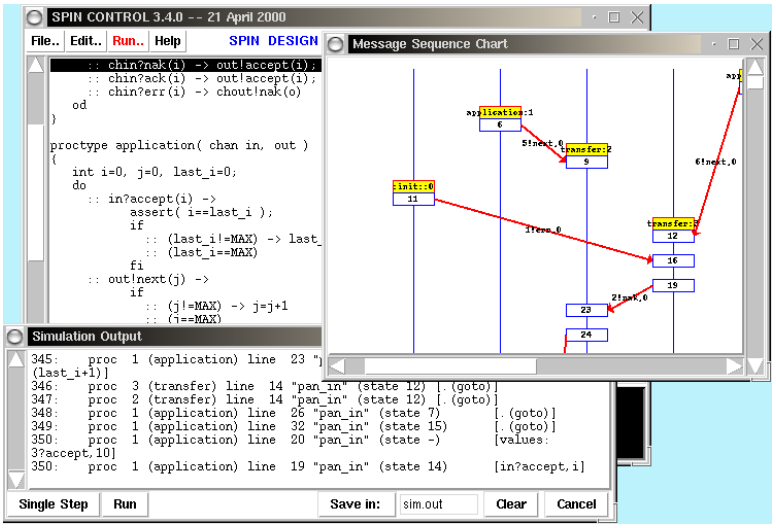
Perl in web page developments, Tcl/Tk for GUI development

How not to!

Don't use to the exclusion of other languages!

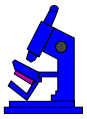
- ✔ Good and bad
- ✔ Array lookup ..
- ✔ Common to mix scripting and other languages.

Xspin



Hello world

```
manu> wish
wish> button .quit -text "Hello World!" -
command {exit}
.quit
wish> pack .quit
wish>
```



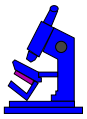
In a script

CODE LISTING **HelloWorld.tcl**

```
#!/usr/local/bin/wish8.1 -f

button .quit -text "Hello World!" -command {exit}
pack .quit
```

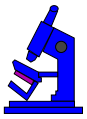

Run it



Tcl first...then Tk

Tcl structure

- ✔ Tiny syntax
- ✔ Single *command* structure
- ✔ Set of rules
- ✔ All PROGRAM structures are implemented as *commands*.
- ✔ Level' of the GUI interface is quite high.



Comments

If the first character of a command is #, it is a comment.

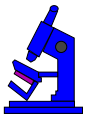
Commands

Tcl commands are just words separated by spaces. Commands return strings, and arguments are just further words.

```
command argument argument
command argument
```

Tcl commands are separated by a new line, or a semicolon, and arrays are indexed by text:

```
set a(a\ text\ index) 4
```



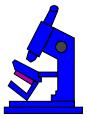
Command examples

Procedures	File Access	Miscellaneous
`proc name {params} {body}`	`open <name>`	`source <NameOfFile>`
	`read <fileID>`	`global <varname>`
	`close <fileID>`	`catch <command>`
	`cd <directoryname>`	`format <fmtstrng> <val>`
		`exec <process>`
		`return <value>`

Spaces

Spaces are important:

```
expr 5*3       has a single argument
expr 5 * 3     has three arguments
```



Tcl/Tk quoting rules

When the " or { character are first in the word.

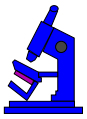
".." disables a few of the special characters - for example space, tab, newline and semicolon, and

{..} disables everything except \{, \} and \nl.

Control structures

End up looking very like 'C':

```
while {a==10} {
   set b [tst a]
}
```



Command

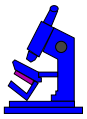
The []'s are replaced by the value returned by executing the Tcl command 'doit'.

```
set a [doit param1 param2]
```


Variable substitution

The dollar sign performs the variable value substitution. Tcl variables are strings.

```
set a 12b     a will be "12b"
set b 12$a    b will be "1212b"
```

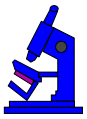


Backslash

```
set a a\ string\ with\ spaces\ \
and\ a\ new\ line
```


tclsh or wish?

- ✔ tclsh - a shell-like application that reads Tcl commands from its *standard input* or from a *file* and evaluates them.
- ✔ wish - the Tcl command language, the Tk toolkit, and a main program that reads commands from standard input or from a file.



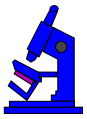

Assignment

```
% set a 1
1
% set a
1
% set a 2
2
% set a
2
```



Assignment

```
% set a alhabetapruning
alhabetapruning
% set a
alhabetapruning
% set a alpha\ beta\ pruning
alpha beta pruning
% set a
alpha beta pruning
```



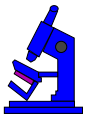
Assignment

```
% set a alhabetapruning
alhabetapruning
% set a
alhabetapruning
% set a alpha\ beta\ pruning
alpha beta pruning
% set a
alpha beta pruning
```

(Backslash substitution)

Assignment

```
% set a {alha
beta
pruning}
....
% set a
alpha
beta
pruning
```



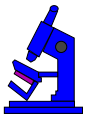
Assignment

```
% set a 22/3
22/3
% set a
22/3
```


Assignment

```
% set a expr 22/3
wrong # args: should be "set varName ?newValue?"
% set a {expr 12/4.2}
expr 12/4.2
% set a [expr 12.2/33]
0.369696969697
% set a
0.369696969697
```

(Command substitution)



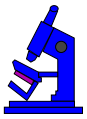
Assignment

```
% set a
0.369696969697
% set a [expr $a*33]
12.2
```

(Variable and Command substitution)

Assignment

```
% expr 1+2+3
6
% expr 1 + 2+3
6
```



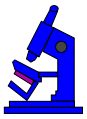
Assignment - increment

```
% set a 10
10
% incr a
11
% set a
11
```


Assignment

```
% set g gallon
gallon
% set gallons $gs
can't read "gs": no such variable
% set gallons ${g}s
gallons
```

(Variable substitution)



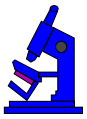
Assignment - append

```
% set a ${a}222
11222
% append a 333
11222333
```


Tcl/Tk list

Just a sequence of words:

```
% set dow {Mon Tue Wed Thu Fri Party Sun}
Mon Tue Wed Thu Fri Party Sun
% lindex $dow 3
Thu
%
lindex $dow 1
Tue
```

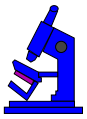


Lists within lists

```
% set a {0 1 {2 x} {3 x y} 4}
0 1 {2 x} {3 x y} 4
% lindex $a 3
3 x y
```


Iteration over list

```
% foreach day $dow {
     puts Day\ of\ week\ is\ $day
  }
Day of week is Mon
Day of week is Tue
Day of week is Wed
Day of week is Thu
Day of week is Fri
Day of week is Party
Day of week is Sun
```



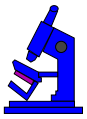

Arrays

```
% set work(Mon) 8
8
% set work(Tue) 10
10
% foreach day $dow {
     puts $day\ I\ worked\ $work($day)hrs
  }
Mon I worked 8hrs
Tue I worked 10hrs
can't read "work(Wed)": no such element in array
```



Arrays

```
% array size work
2
% array names work
Tue Mon
```



Summary

- ✔ Tcl
  - ✔ commands
  - ✔ quoting and substitutions
  - ✔ assignment (set)
  - ✔ Lists and arrays