An Overview of Tcl and Tk

John Ousterhout
Sun Microsystems Laboratories
`john.ousterhout@eng.sun.com`

Tcl/Tk Tutorial, Part I

Introduction

- **Component technologies:**
  - Tcl: embeddable scripting language
  - Tk: GUI toolkit and widgets based on Tcl.
- **The principle: universal scripting language controls everything: functions, interfaces, communication.**
- **Results:**
  - Raise the level of X programming: simpler, 5-10x faster application development.
  - Greater power: more things programmable, applications work together.
  - Active objects: replace data with scripts.

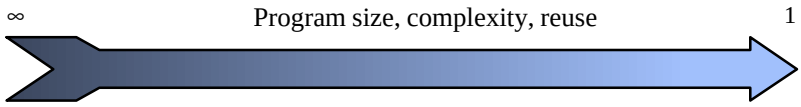
Outline

- **Tcl scripting language.**
- **Tk toolkit.**
- **Tk applications.**
- **Survey of applications and extensions.**
- **Conclusions.**

Tcl: Tool Command Language

- **Interactive programs need command languages:**
  - Typically redone for each application.
  - Result: weak, quirky.
  - **`emacs`** and **`csh`** powerful, but can't reuse.
- **Solution: reusable scripting language.**
  - Interpreter is a C library.
  - Provides basic features: variables, procedures, etc.
  - Applications extend with additional features.

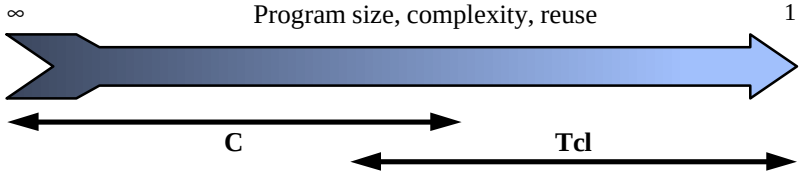
Scripting Language Philosophy



- **Large, complex applications:**
  - Performance important.
  - Need structure.
  - Goal: prevent bad things.
- **Interactive commands, scripting:**
  - Performance less important.
  - Minimum structure: less overhead, easy interchange.
  - Goal: enable good things.

One language can't meet all needs?

Two-Language Approach



- **Use Tcl for scripting, C or C++ for large things.**
- **Goals for Tcl:**
  - Minimal syntax: easy to learn and type.
  - Minimal structure: make things play together.
  - Simple interfaces to C: extensibility.

Tcl: Tool Command Language

- **Simple syntax (similar to sh, C, Lisp):**

```
set a 47                ⇨ 47
```

- **Substitutions:**

```
set b $a                ⇨ 47
set b [expr $a+10]      ⇨ 57
```

- **Quoting:**

```
set b "a is $a"         ⇨ a is 47
set b {[expr $a+10]} ⇨ [expr $a+10]
```


More On The Tcl Language

- **Rich set of built-in commands:**
  - Variables, associative arrays, lists.
  - C-like expressions.
  - Conditionals, looping:

```
if "$x < 3" {
    puts "x is too small"
}
```

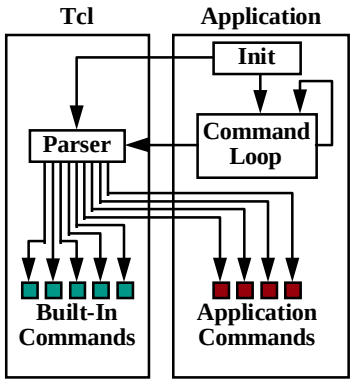
  - Procedures.
  - Access to UNIX files, subprocesses.
- **Only representation is strings:**
  - Easy access from C.
  - Programs and data interchangeable.

Factorial Procedure

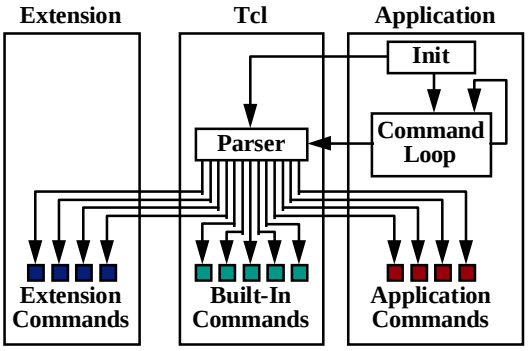
```
proc fac x {
    if $x<=1 {return 1}
    expr $x*[fac [expr $x-1]]
}

fac 4   ⇨ 24
```


Embedding Tcl In Applications

- **Application generates scripts.**
- **Tcl parses scripts, passes words to command procedures.**
- **Application extends built-in command set:**
  - Define new object types in C.
  - Implement primitive operations as new Tcl commands.
  - Build complex features with Tcl scripts.



Extensions



- **Extensions can be developed independently:**
  - Network communication, database access, security, ...
- **Applications can include combinations of extensions.**

The Tk Toolkit

- **The problem:**
  - Too hard to build applications with nice user interfaces.
- **The wrong solution:**
  - C++, object-oriented toolkits.
  - Only small improvement (10-20%?): must still program at a low level.
- **The right solution:**
  - Raise the level of programming.
  - Create interfaces by writing Tcl scripts.

Creating User Interfaces With Tk

- **Additional Tcl commands:**
  - Create Motif-like widgets.
  - Arrange widgets.
  - Bind events to Tcl commands.
  - Manipulate X selection, focus, window manager, etc.
- **Library of C procedures:**
  - Create new widget classes.
  - Create new geometry managers.

What's A Tk-Based Application?

- **The Tcl interpreter.**
- **The Tk toolkit.**
- **Application-specific C code (primitives!):**
  - New object types.
  - New widgets.

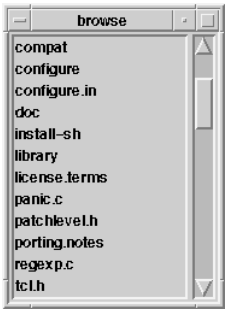
  } Tcl commands
- **Tcl scripts (compose primitives):**
  - Build user interface.
  - Respond to events.

Wish: Windowing Shell

- **Create user interfaces by writing Tcl scripts.**
- **Hello, world:**

```
button .hello -text "Hello, world" -command exit
pack .hello
```

- **Simple directory browser: 30 lines**
- **Web browser: 2000 lines**
- **10x less code for simple things.**



Browser Wish Script

```
#!/usr/local/bin/wish4.0

listbox .list -yscroll ".scroll set" \
    -width 20 -height 20
pack .list -side left
scrollbar .scroll -command \
    ".list yview"
pack .scroll -side right -fill y

if {$argc > 0} {
    set dir [lindex $argv 0]
} else {
    set dir .
}
foreach i [lsort [glob * .*]] {
    .list insert end $i
}
```

```
bind .list <Double-ButtonPress-1> {
    browse $dir [selection get]
}
bind all <Control-c> {destroy .}

proc browse {dir file} {
    if {$dir != "."} {
        set file $dir/$file
    }
    if {file isdirectory $file] {
        exec browse $file &
    } else {
        if [file isfile $file] {
            exec xedit $file &
        } else{
            error "can't browse $file"
        }
    }
}
```

Other Uses For Tcl

- **Data representation:**
  - Store data on disk as Tcl scripts.
  - To load information, evaluate script.
- **Active objects:**
  - Store scripts in objects.
  - Evaluate scripts at interesting times to give objects behavior.
- **Communication: send Tcl scripts between applications.**
- **Executable content:**
  - Active e-mail messages, Web pages.
  - Agents.

Status

- **Runs on all UNIX/X platforms.**
- **PC/Mac alpha releases: September 1995.**
- **Source and documentation freely available.**
- **100,000 developers world-wide?**
- **Hundreds of commercial products, free extensions.**
- **Newsgroup: comp.lang.tcl.**
- **2 introductory books (Ousterhout, Welch).**

Representative Applications

- **Multimedia, groupware.**
- **Active e-mail messages.**
- **System administration.**
- **Testing.**
- **Scientific applications:** instrument control, simulation, visualization, CAD.
- **Real-time control system for offshore platform.**
- **British teletext system.**
- **Feature animation at Walt Disney Studios.**
- **On-air broadcast control system for NBC.**

Popular Extensions

- **Available via public FTP:**

  `ftp://ftp.aud.alcatel.com/tcl`

- **Expect: remote control for interactive programs such as `ftp`, `telnet`, `crypt`, and `fsck`:**

```
#!/usr/local/bin/expect
spawn rlogin [lindex $argv 0]
expect -re "($|#|%) "
send "cd [pwd]\r"
expect -re "($|#|%) "
send "setenv DISPLAY $env(DISPLAY)\r"
interact
```

Popular Extensions, cont'd

- **TclX: general-purpose extensions:**
  - POSIX system calls.
  - Keyed lists.
  - File scanning (similar to awk).
  - Date/time manipulation.
  - Debugging, help, profiling.
- **Oratcl and Sybtcl: access to commercial databases.**
- **Incr tcl: object-oriented programming in Tcl.**

Popular Extensions, cont'd

- **Tcl-DP: socket-based remote procedure calls, distributed objects.**
  - Sample server:

```
set myId 0
proc GetId {} {
    global myId
    incr myId
    return $myId
}
dp_MakeRPCServer 4545
```

  - Sample client:

```
set server [dp_MakeRPCClient foo.bar.com 4545]
dp_rpc $server GetId
```

Where You Might Use Tcl and Tk

- **Creating graphical user interfaces.**
- **Testing.**
- **Applications that need scripting or extension facilities.**
- **Platform-independent applications.**

Drawbacks

- **Must learn new language (substitution rules confusing to some people).**
- **Interpreted language has performance limits (but surprisingly high).**
- **C interfaces incompatible with Xt, Motif library.**

Plans For The Future

- **Increase accessibility:**
  - More work on PC/Mac ports (native look and feel).
  - SpecTcl: interactive GUI builder.
  - Better development tools (debugger, etc.)
- **Improve the language/toolkit:**
  - Incremental on-the-fly compiler.
  - Better support for modules, data structures.
  - Better internationalization (Asian character sets).
- **Create exciting Internet applications:**
  - Active documents?

Conclusions

- **High-level programming:**
  - Less to learn.
  - Build applications more quickly.
- **Universal scripting language:**
  - Extend and modify applications at run-time.
  - Make many things work together.
- **Use scripts instead of data:**
  - Active objects, executable content.

Tcl + Tk = shell of the 1990's?