Using and Abusing TclLib

Gerald W. Lester
HMS Software, Inc.

Overview

- What is TclLib
- Tour of Modules
- Some Short Examples
  - Parallel Difference Utility
  - Sending E-mail with Attachments
  - FTP Mirror
- How you can help

What is TclLib

- Collection of “pure” Tcl modules
  - Several use C extensions if available
    - Provides the same functionality
    - Provides performance improvement
    - One interface for programmer
- “Certified” modules
  - Test cases
  - Documentation
- Open Source
  - http://tcllib.sf.net
- All code under BSD license
  - Same as Tcl/Tk
  - Can be used in commercial applications

Overview of Modules

- Programming Tools
- Mathematics
- Data Structures
- Networking
- CGI Programming
- Hashes, Checksums and Encryption
- Text Processing
- Documentation Tools
- File Format
- Grammars and Finite Automata

Programming Tools

- Two Object Systems
  - SNIT
  - STOOP
- Logging utilities (log and logger)
- Tcl Source Code Profiler
- Persistence array (tie)
- One-to-many communication with sockets
- Control flow constructs
- File Utilities

Snit vs Stoop

- Snit
  - Based on
    - Components
    - Delegation
    - Not inheritance
  - Primary purpose is to be object glue
    - To compose diverse objects from diverse sources into types and megawidgets
  - Efficient as hand coded Tcl objects
- Stoop
  - More C++ or [Incr Tcl] like
  - A major design consideration was to have minimum adverse impact on performance

Mathematics

- High Precision Constants
  - Pi, e, etc
- Fuzzy comparison of floats
- Complex Numbers
- Interpolation
- Integration
- Polynomial math and evaluation

Data Structures

- Create and maipulate
  - Lists
  - Sets
  - Stacks
  - Queues
  - Priority Queues
  - Trees
- Create and maipulate
  - Graphs
  - Records
  - Matrix
  - Reports on Matrix
  - Pools
  - Skip List
    - An alternative to binary trees

Networking

- Client only
  - DNS
  - LDAP
  - NTP (time)
  - NNTP (news)
  - IRC
  - IDENT (RFC 1413)
- Client and Server
  - FTP
  - POP3 (receive mail)
    - Includes simple user DB and mailbox server impelentation
  - SMTP (sending mail)
    - Uses MIME

Networking

- Utitlites
  - Ipv4 and IPv6
  - URI
  - Autoproxy
- Encoder/Decoders
  - BitTorrent Serialization Format
  - ANS.1 BER

CGI Programming

- Generation of Pages
  - HTML
  - JavaScript
- Reading of Forms
  - ncgi
    - Not to be confused with Don Libes CGI package

Text Processing and File Formats

- Encoding/Decoding
  - Base64, uuencode, Yencode
- File Formats
  - CSV, Windows INI, JPEG, PNG, EXIF fields from digital images
- Other Formats
  - HTML parser, MIME
- General Text Utility
  - Template processing
  - String vs Character “extensions” of Tcl Commands
    - Split, Trim, (un)Tabification, (de)Indent

Documentation Tools

- DocTools
  - Text with embedded commands
    - Commands are in “[“ “]”
  - Engines to generate different targets from same source
  - Table of Contents and Index support

Hashes, Checksums and Encryption

- Check Sums
  - Cksum(1), Sum(1), CRC16, CRC32
- Hashes
  - SoundEx, sha1, md4, md5, RIPEMD-128, RIPEMD-160, md5crypt
- Encryption/Decryption
  - DES
- Universally Unique Identifiers

Grammars and Finite Automata

- Create, manipulate and execute Finite State Automatons
  - When executing callback gets invoked if an error, reset or final state is entered

Example 1 – Parallel Difference

Puts out a list of lines consisting of:

n1<TAB>n2<TAB>line

where n1 is a line number in the first file, and n2 is a line number in the second file. The line is the text of the line. If a line appears in the first file but not the second, n2 is omitted, and conversely, if it appears in the second file but not the first, n1 is omitted.

Usage: file1 file2

Example 1 - Parallel Difference

```
package require struct

##
## Open the files and read them into memory
##
foreach fn {0 1} {
    set fd [open [lindex $argv $fn] r]
    set lines($fn) [split [read $fd] \n]
    close $fd
}

set i 0; set j 0;

##
## Do the real work
##
::struct::list assign \
    [::struct::list longestCommonSubsequence $lines(0) $lines(1)] \
    x1 x2
```

Example 1 - Parallel Difference

```
##
## Output until one file runs out
##
foreach p $x1 q $x2 {
    ## Output lines in file 1 but not 2
    while { $i < $p } {
        set l [lindex $lines(0) $i]
        puts "[incr i]\t\t$l"
    }
    ## Output lines in file 2 but not 1
    while { $j < $q } {
        set m [lindex $lines(1) $j]
        puts "\t[incr j]\t$m"
    }
    ## Output lines in both files
    set l [lindex $lines(0) $i]
    puts "[incr i]\t[incr j]\t$l"
}
```

Example 1 - Parallel Difference

```
##
## Output remaining lines in file 1
##
while { $i < [llength $lines1] } {
    set l [lindex $lines1 $i]
    puts "[incr i]\t\t$l"
}

##
## Output remaining lines in file 2
##
while { $j < [llength $lines2] } {
    set m [lindex $lines2 $j]
    puts "\t[incr j]\t$m"
}
```

Example 2 - Sending Mail

- Send mail with attachments

```
package require smtp
package require mime

##
## Contents are hard coded - modify as you like
##
set server stmp.nowhere.com
set toList {p.krum@redneck.edu l.eshkin@papermill.edu}
set fromList {i.asimov@nyu.edu}
set subject {Purity of Thiotimoline Sample}
set body {blah blah blah}
set attList {chart1.pdf chart2.pdf}
array set attTypes {
    .pdf {application/pdf}
}
```

Example 2 - Sending Mail

```
##
## Create body
##
set partsList [::mime::initialize \
                   -canonical text/plain \
                   -string $body]

##
## Create attachments
##
foreach attachment $attList {
    set ext [file extension $attachment]
    lappend partsList [::mime::initialize \
                           -canonical $attTypes($ext) \
                           -file $ext]
}
##
## Create main message
##
set messageToken  [::mime::initialize \
                       -canonical multipart/mixed \
```

Example 2 - Sending Mail

```
##
## Create main message
##
set messageToken  [::mime::initialize \
                       -canonical multipart/mixed \
                       -parts $partsList]

##
## Send it
##
::smtp::sendmessage $messageToken \
    -servers $server \
    -header [list TO toList] \
    -header [list From $fromList] \
    -header [list Subject $subject]
::mime::finalize $messageToken -subordinates all
```

Example 3 - FTP Mirror

Mirrors a remote directory tree locally

```
package require ftp 2.0

##
## Configuration is hard coded - put your own UI
##
set server noname
set username anonymous
set passwd xxxxxx
set dirFmt "%s %s %s %s %s %s %s %s %s %s %s"

##
## Simple progress display, put a “.” out every time a
## block of data is transferred
##
proc ProgressBar {bytes} {
    puts -nonewline stdout "."; flush stdout
}
```

Example 3 - FTP Mirror

```
##
## Recursive file transfer
##
proc GetTree {conn {dir ""}} {
    catch {file mkdir $dir}
    foreach line [ftp::List $conn $dir] {
       set rc [scan $line $::dirFmt \
            perm l u g size d1 d2 d3 name link lnksrc]
       if {[string equal $name "."] ||
           [string equal $name ".."]} {
          continue
       }
       set type [string range $perm 0 0]
       set name [file join $dir $name]
       switch -- $type {
           d {GetTree $name}
           l {catch {file link -symbolic $lnksrc $name}}
           - {ftp::Get $conn $name}
       }
    }
}
```

Example 3 - FTP Mirror

```
##
## Main
##

##
## Open the connection
##
set conn [ftp::Open $server $username $passwd \
              -progress ProgressBar]

##
## If we made the connection, then do the work!
##
if {$conn != -1} {
    GetTree $conn
    ftp::Close $conn
    puts "OK!"
}
```

How Can I Help?

- Contribute
  - Code
    - For anything you think is missing
    - For missing protocols such as IMAP
  - Better documentation
  - More examples
- Help maintain a module
- File a bug report if you find something wrong
- Tell others about TclLib

Summary

- Be Lazy
  - Use TclLib
  - Don't reinvent the wheel
    - Lots of functionality with very little code
- Pure Tcl Modules
  - But takes advantage of C extension if present
- Good, Clean, Safe Code
- Tcl License
  - Can be used in commercial applications

Questions?