Porting tcllib to Tcl 9

Report on a work in progess

Rolf Ade

Motivation

- It’s needed
  - without the eco-system Tcl 9 will not fly
- To learn what it means to port a lager code base
- To have a larger code base to hammer on the Tcl 9 core

Some Notes in Front

- tcllib is special
  - lots of pkgIndex.tcl files
  - lots of `packages require Tcl 8.X`
- Just pure Tcl so far, no critical parts
- Most packages have test suites
  - some noise by test constraints
    and changed expectations

Status

- All test suits run without errors with Tcl 8.5 / 8.6 / 9 on linux (tcllib repository branch tcl9-testarea-rde)
- Still work in progress – to do:
  - tilde change (on the way)
  - “Build“ / Installation tests
  - Windows / Mac tests
  - closer look upon the core running under valgrind
  - review after default I/O profile decision
  - Maybe compatibility package to smooth a few things

Findings (so far)

- Typically little, if any changes needed on script level
  - most binary extensions must be touched (API changes)
  - most scripts run unchanged (minus trivial changes as `package require 8.5-`)
- Lots of the needed “real“ changes are easy to spot and fix
- Although, there **are** hard to find / “tricky“ things (see later)
  - helpful to know the code base
  - therefore porting a legacy code base may need care / work
- Work reports like this will help you migrating your code
- The Tcl 9 core proofed to be robust, so far

Obvious Things I

The package system

Replace

```
package require Tcl 8.5
```

with

```
package require Tcl 8.5-
```

and

```
package vsatisfies [package provide Tcl] 8.6
```

with

```
package vsatisfies [package provide Tcl] 8.6 9
```

or

```
package vsatisfies [package provide Tcl] 8.6-
```

Requires at least Tcl 8.5

Obvious Things II

- No `tcl_precision` anymore (TIP 488)

  Just remove

- `string bytelength` is gone in Tcl 9 (TIP 597)

  Replace it with:
  ```
  string length [encoding convertto utf-8 $string]
  ```

- Underscore is now allowed in (not at start) a number (TIP 551)

  ```
  string is integer 123_456 => 1
  ```

  Check if you want this!

Obvious Things III

Variable resolution (TIP 278)
In namespace code (example from crc/crc32.tcl):

```
if {[info exists tcl_platform(wordSize)]} {
    set signbit [expr {1 << (8*$tcl_platform(wordSize)-1)}]
}
```

Change to:

```
if {[info exists ::tcl_platform(wordSize)]} {
    set signbit [expr {1 << (8*$::tcl_platform(wordSize)-1)}]
}
```

Can be a tricky to find

Obvious Things IV

Tilde isn’t special at the start of file path anymore (TIP 602)

Didn’t pop up in any tcllib test (!)

But it is there. Work in progress.

Code which should run with Tcl 8.[56] and Tcl 9 needs something like:

```
if {[package vsatisfies [package present Tcl] 9]} {
    set pd [file join [file home] .[join $prefix /] plugin]
} else {
    set pd [file join ~ .[join $prefix /] plugin]
}
```

Obvious Things V

Octal by leading zero is gone in Tcl 9 (TIP 114)

Grep for octal numbers
Make them explicit by `0o`

Example (from base64/uuencode.tcl):

```
append r [Enc [expr {($c3 & 077)}]]
```

Change to:

```
append r [Enc [expr {($c3 & 0o077)}]]
```

Not so obvious Things I

- `string is integer` chanced due to octal died

  Tcl 8.6:
  `string is integer 09` => `0`

  Tcl 9.0:
  `string is integer 09` => `1`

- Error dict changes (no big surprise)

  What you get from

  ```
  catch {script) result optionVarName
  ```

Not so obvious Things II

`binary scan` and `binary encode` raise error in Tcl 9 on non-bytes (TIP 568)

Tcl 8:
`binary scan \uabcd c* result` => 1 (with result -51)

Tcl 9:
`binary scan \uabcd c* result` => error ("expected byte sequence but character 0 was 'ꯍ' (U+00ABCD)")

From uuid/uuid.tcl:

```
set fin [open /dev/urandom r]  <=== add „b“
binary scan [read $fin 128] H* machinfo
close $fin
```

Not so obvious Things III

- `expr int()` changed (TIP 514)

  int() does not do either 32-bit or 64-bit truncation anymore

- Octal is dead, again

  Tcl 8:

  `file attributes $name -permissions` ==> something like **00**644

  Tcl 9:

  `file attributes $name -permissions` ==> something like **0o**644

  Example: tar/tar.tcl – writing a value in file format specific octal representation

Not so obvious Things IV

Octal is dead, again

Look for

`format %o $value` ==> result does not start with 0o

Every value meant be octal can be tricky.

Example from pt/char.tcl (parse/detect octal input in grammer):

```
if {[regexp {^\\([0-2][0-7][0-7])$} $ch -> ocode]} {
    return [format %c $ocode]
}
```

Change to:

```
if {[regexp {^\\([0-2][0-7][0-7])$} $ch -> ocode]} {
    return [format %c 0o$ocode]
}
```

Not so obvious Things V

Variable substitution slightly changed (TIP 465)

Tcl 8:

`set map(() foo; puts $map(()` => foo

Tcl 9:

`set map(() foo; puts $map(()` => error (invalid character in array index)

Happens in the wild (example from ncgi.tcl):

```
if {![string match \[a-zA-Z0-9\] $c]} {
    set map($c) %[format %.2X $i
}
```

(Still open for other reasons, see https://core.tcl-lang.org/tcllib/tktview/1f29f7baf9)

Finis.

Questions?