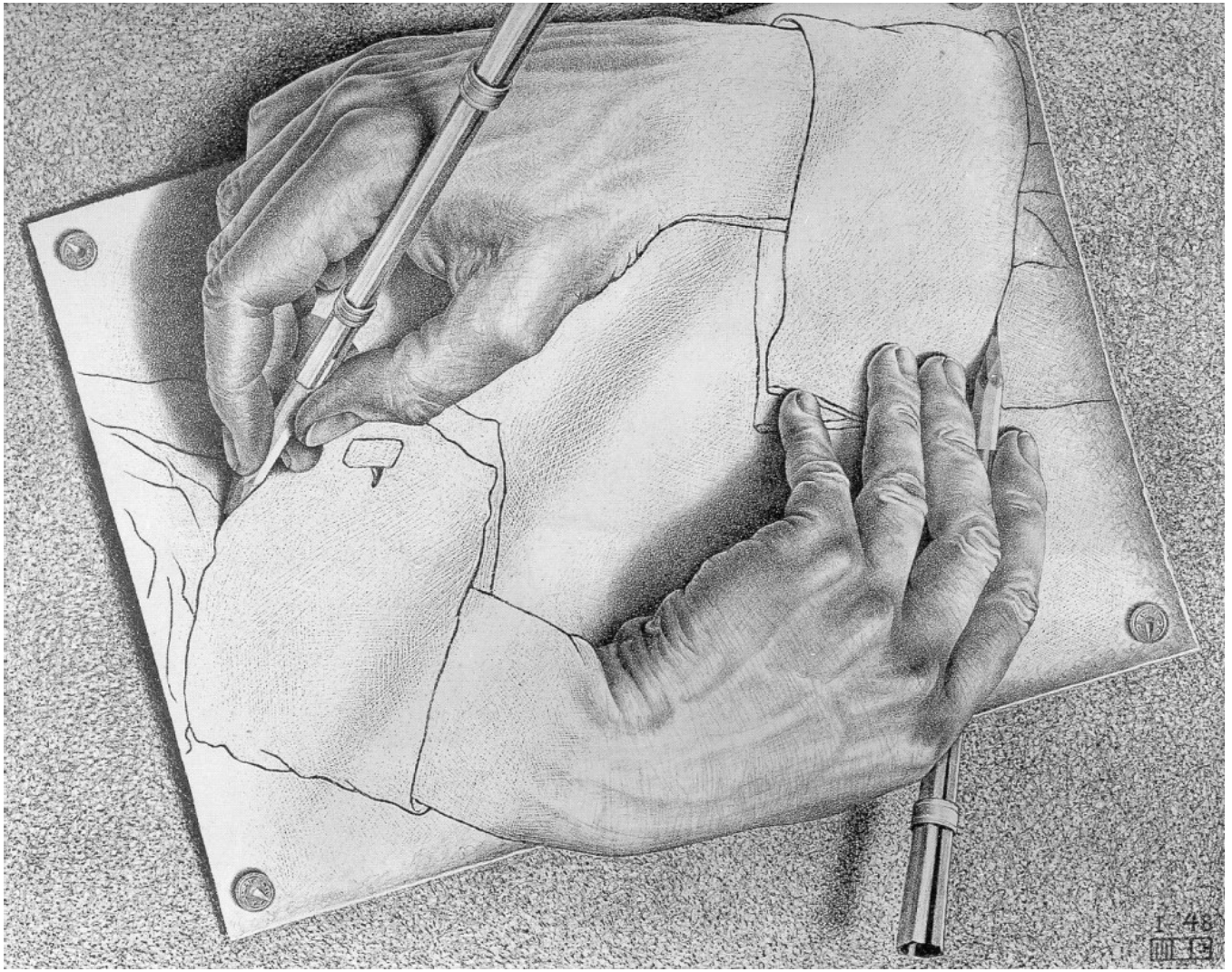
Practcl

Package Repository Automation for C and Tcl



URLS

- Latest Draft of the Paper:

  http://www.etoyoc.com/tcl/Practcl.pdf

- This Presentation:
  - http://www.etoyoc.com/tcl/Practl_Paper_Presentation.odp
  - http://www.etoyoc.com/tcl/Practl_Paper_Presentation.pdf

Problem Statement:

- Every large program is its own Tcl Distribution
  - Examples: BRLCad, VTK, Androwish
- There is no “community standard” automation for building Tcl Distributions
- Virtually every major project has a different strategy to build their final product
- Different platforms require customizations that aren't captured and shared between projects

Problem Statement

- In the end, every large Tcl/Tk application is it's own Tcl distribution
- Every distro maintainer also has to bug fix or modernize staleware Tcl extensions

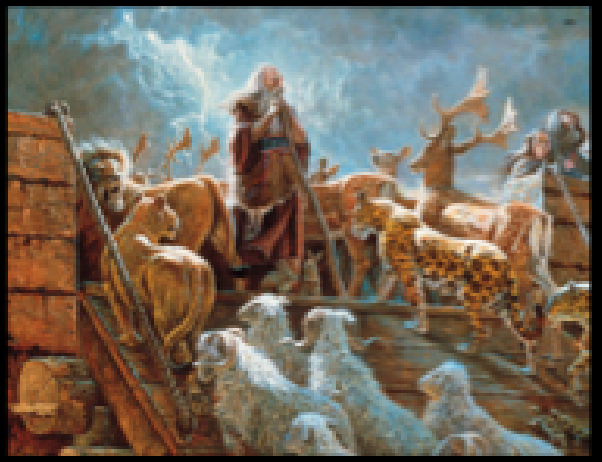
Tcl Distro Maintainer

What my friends thinks I do

What my wife thinks I do

What the tcl community thinks I do

What my users think I do

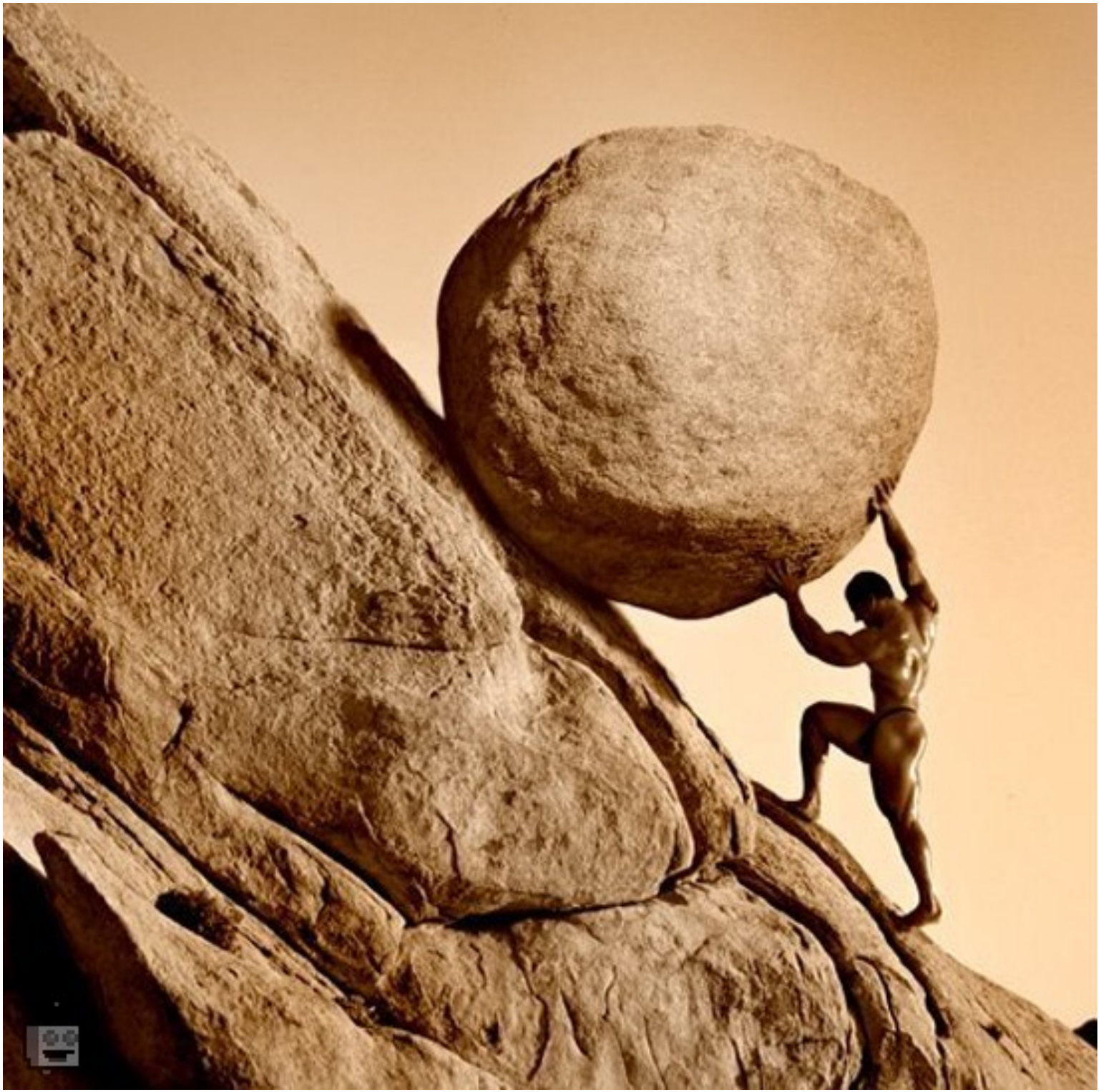


What I think I do

What I really do

Mad Scientist

- To someone unfamiliar with the process this is pure mad-scientist grade stuff.
- Possibly crazy, possibly genius



Pushing a Rock Uphill

- To most Tcl/Tk users, this seems like it should be a “solved problem”
- Except that the solutions presently are ugly and “works on my workstation” specific

Hamster On a Wheel

- To the users of my application, it always seems like I'm pushing new versions that don't really change anything
- And when things to change… it's because the wheel has fallen off

I Feel like Noah

- I have to build my own archives of things for a rainy day
- Which makes me look like a frigging lunatic…
- ...until it actually rains

My Cunning Plan...

- ...occasionally has shocking results when I try to connect things in previously untested ways

Why is maintaining a Tcl Distro so hard?

- Tcl has an enormously tolerant API
- Many extensions can continue to live on a decade after their developers have lost interest in maintaining them
- There have been several “great ideas” on how to make Tcl extensions over time

A Brief History of Tcl

- Tcl began as an embed-able scripting language for larger applications
- Tcl Extended programs, not the other way around

Early Tcl Extensions

- Grab a copy of the Tcl/Tk sources
- Copy public distributed extensions
- Add new functions
- Distribute your executable/library

myprog

tcl

tk

tclx (and other extensions)

myprog_internals

Pros/Cons

- Any C programmer could write a Tcl/Tk application
- Adding new functions is trivial

- Upstream changes to the Tcl core could lead to massive rewrites in derivative works
- Duplication of effort
- Key useful functions had radically different implementations

Put More in the Core

- The core team (Dr. Ousterhout and later Sun and Scriptics) would bang together new editions of the Tcl/Tk core every 6-12 months
- New features were contributed by the community, with final implementation by the core team

Pros/Cons

- Tcl picked up a lot of general purpose functions
- It's easier to code in C than hack in autoconf/Make

- Core Developers had to review and reimplement every extension of them
- Non “core” extensions still had to track with changes in the core
- ...and the other extensions

TEA

- Starting in 1999 an effort was made to standardized build systems and the Tcl API
- 8.1 introduced stubs
  - Extensions were no longer tied to a specific tcl version
  - Minimized the need for access to the Tcl internals
- Scriptics developed TEA: the Tcl Extension Architecture
  - Compiles Extensions for Unix, Window, and Mac

Features of TEA

- It was familiar to Tcl users of the day
- Extensions were built as a microcosm of the core build system
- The same autoconf (and later nmake) systems in the core were replicated into TEA
- The core and extensions would publish their API as part of the build process

Pros/Cons

- Building a Tcl extension was exactly like building the Tcl core
- Plenty of shims for developers to modify the build system to suit their needs
- Built around autoconf, a dependible, portable...

- Building a Tcl extension was *exactly* like building the Tcl core
- Plenty of rope for developers to hang themselves with
- ...and unix-only shell script

Pros/Cons

- Tcl supported both Unix and Windows style build automations...
- ...independently

Critcl

- A build system for Tcl extensions
- Runs inside Tcl
- Can extend Tcl on the fly
- Can build its own wrapped executable from its own source

Pros/Cons

- Fantastic for developing new extensions
- No help at all for updating existing extensions

Wait… this was about maintaining Tcl Distributions

- In the course of keeping my software up to date, I end up having to keep other people's software up to date

Sean's Island of Misfit Toys:

- Bits of Tcllib
- Canvas3D
- Tclconfig (Aka: “TEA”)
- TclUDP
- TkHtml
- ZipFS (The C implementation)

What is Practcl?

- Practcl is a library of re-usable tools needed for package building and Tcl distribution maintenance
- It is designed for maintaining simultaneous releases for Windows, Macos, and Debian Linux
  - (AKA: The platforms I use on a regular basis and can physically test on.)

Practcl Includes:

- Several enhancements to the tcl.m4 file to support non-native Window systems
  - SDL
  - X11 on Macos
- Several enhancements for building TEA extensions as a static library
  - Stubs are now a configuration option… that works

Practcl Introduces:

- A world without GnuMake or Windows Batch files
- Build automation is now in Tcl
  - Compilers and linkers are called via Exec
  - A new config.tcl file renders autoconf substitutions in a Tcl-readable format
  - The build system can operate and interact with a toplevel automation prior to running ./configure
  - Developers can invest their time battling complexity in a language they are familiar with: Tcl

Yummy Yummy Carrots

- Includes facilities for building wrapped executables (Using ZipFS)
- One Tcl Script can perform a complete automated build or an entire distribution
  - ***Pulling Source from Git/Fossil for specifically tagged versions***
  - ***Building the Tcl Core… statically***
  - ***Building Tcl Extensions either statically or dynamically***
  - ***Cargo culting binaries where building is difficult or dangerous***
  - ***Cross compiling (and keeping all of the build flags consistent)***
  - ***Installing build tools into the local environment***
    - ***Critcl***
    - ***Tcllib***

- (And it even does it all in the right order)
- (And it still uses the same autoconf that TEA did, only using Tcl to pass the commands instead of Make/Nmake)

How To Use (For extensions)

- Practcl is distributed side-by-side with the tcl.m4 in the tclconfig distribution
- Developers add a “make.tcl” file, and redirect parts of the installation process to exercise that script
- Eventually the Makefile is a hollowed out husk which is simply offers backward compatible redirects to the make.tcl script
  - We preserve that so that the “./configure ; make ; make install” mantra can carry forth

How To Use (For Distributors)

- Create a “make.tcl” file for your distribution
- Include a copy of practcl.tcl in your toplevel automation
- List what packages you want included
- If cross compiling, prepare a “config.site” file
- State which extensions are to be part of your basekit, and which will be installed in the environment or within your application's VFS
- Grab coffee

Demos

- Dang… 37 slides in… I'm probably out of time…
- We'll do a WIP session

Future Applications

- Tip#453
  - Replaces a nasty bit of thrice copied Makefileage with a Tcl Script
  - Adds support for non-TEA extensions in the /pkgs directory of the Tcl sources
  - Powered by Practcl

What's on the Inside

- All functions operate inside of the ::practcl namespace
- Large scale integration is achieved by using TclOO objects
- Facilities are present (and inviting) for projects to create one-off modifications to support custom software

Where To Get it

- Bleeding edge development is in Tclconfig:
  - http://core.tcl.tk/tclconfig
  - In the “practcl” branch
- Releases are mirrored to Tcllib
- If tip#453 comes to pass, a copy will also be distributed with the Tcl Core in 8.7

Where I Need Help

- Practcl was originally intended to allow Windows to build extensions using either MinGW or MSVC.
- I am completely clueless on MSVC
- Linux testing consists of “It works on my machine”
- Unix testing (beyond Macos) is non-existent

Criticisms

- Practcl Requires:
  - a local copy of Tcl 8.5 and TclOO
  - or
  - a local copy of Tcl 8.6 (or better)
- Some big iron distributors build Tcl and its extensions in environments that lack either
  - Or so I've been told
  - By a friend of a friend