# *A Reboot of the Starpack Build Process*

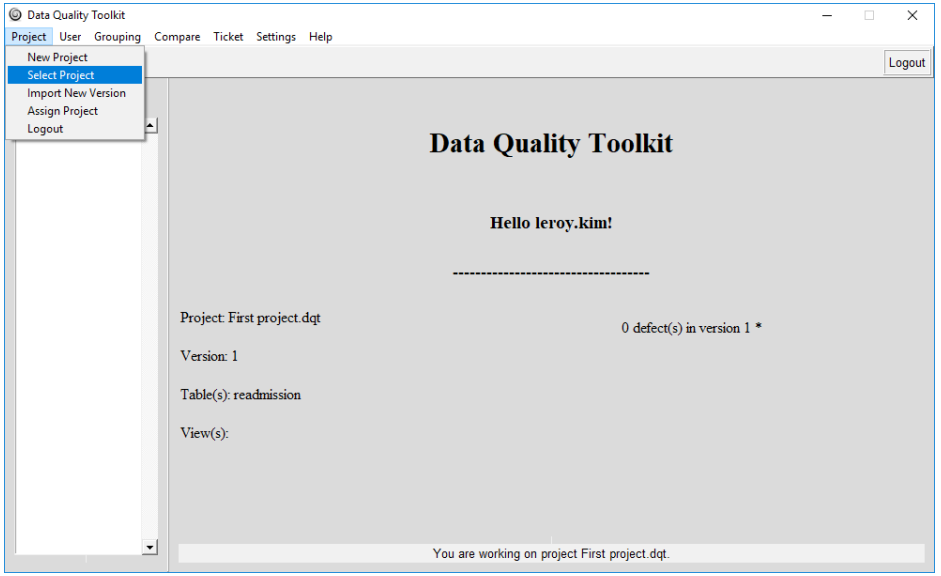
## *For Tcl 8.6 and beyond*

Steve Huntley
Health Information Technology Lab
University of Maryland, Baltimore County

Tcl/Tk Conference 2018

- “How can I package my Tcl/Tk application into a single distributable executable file?”
- “What’s a Starpack”
- “Why do I need a basekit?”
- “What’s a basekit?”
- “Where do I get a basekit?”
- “Is a basekit available for my platform?”
- “My basekit isn’t working.  What do I do?”

# HITLab Data Quality Toolkit (DQT)




- Automatic defect detection in large manually-entered databases
- 470 Tcl code files
- 29 packages
- Compiled loadable shared libraries
- Linux/Windows/???
- Server/client in one deliverable
- Updates after distribution

# ***Motivation:***

Medical ethics:

- safeguard personal information
- accountability, auditability
- results verifiable and reproducible

Engineering:

- transparency
- maintainable
- upgradable
- portable
- stable
- secure

# Basekit Build Tool History

- Tclkit
  - Jean-Claude Wippler (equi4.com)
  - Late 1990's
  - M.sh (1kB shell script)
- Genkit
  - 25kB Tcl script
  - Source code downloaded from equi4.com web site
- Kitgen (2006)
  - config.sh + Makefile

```
#!/bin/sh

# Build TclKit on Linux

V=8.4

P=`pwd`
O=$P/../Dists

cd $O/Tcl/unix
./configure --disable-shared
make libtcl$V.a

cd $O/Tk/unix
./configure --disable-shared --with-
tcl=$O/Tcl/unix
make libtk$V.a

cd $O/Itcl/itcl
./configure --disable-shared --with-
tcl=$O/Tcl/unix
make libitcl3.3.a

cd $O/Mk4tcl/builds
../unix/configure --disable-shared --with-
tcl=$O/Tcl/generic
make libmk4tcl.la

cd $P
pwd

W="-D_LARGEFILE64_SOURCE
-DHAVE_STRUCT_STAT64=1
-DHAVE_TYPE_OFF64_T=1"
D="-DNDEBUG -DKIT_INCLUDES_TK
-DKIT_INCLUDES_ITCL $W"
A="-DTCL_LOCAL_APPINIT=TclKit_AppInit"
I="-I. -I$O/Tcl/generic -I$O/Tk/generic
-I$O/Mk4tcl/include"
L="$O/Tcl/unix/libtcl$V.a
$O/Tk/unix/libtk$V.a \
$O/Itcl/itcl/libitcl3.3.a
$O/Mk4tcl/builds/.libs/libmk4tcl.a"
#X="/usr/X11R6/lib/libX11.a"
X="-L/usr/X11R6/lib -lX11"

rm -f *.o
gcc -c -O3 $I $D $TCL_DEFS src/*.c
$O/Vfs/generic/vfs.c
gcc -c -O3 $I $D $TCL_DEFS $A
$O/Tcl/unix/tclAppInit.c
g++ -static -o kit *.o $L $X -ldl -lieee -lm
-lz

strip kit
rm *.o

rm -f tclkit
./kit

ls -l tclkit
```

# Basekit Build Tool History

- Kitgen Build System (2007)
  - René Zaumseil
  - 100 kB Tcl script
  - Basekit + many library package extensions
  - Shipped with all source code
- Kitcreator (2010)
  - Roy Keene
  - Basekit + many library package extensions
  - 8.6 kB shell script plus extra build scripts per library
  - Source downloaded from respective lib home sites
  - Patches applied to some sources
  - WWWeb interface

# Wanted: new basekit build system

- Minimal *(most desired library extensions can be dynamically loaded as packages from VFS at runtime)*
- Transparent
- Modular
- Configurable
- Extendable
- Hackable
- Fossil repository control → *backup/archive*

# Architecture: changes for Tcl 8.6

- Libraries included in basekit:
  - Tcl
  - TclVFS (virtual filesystem)
  - Mk4tcl (Metakit)
  - zlib
  - rechan (reflected channel)
  - pwb (encoding bug workarounds)
  - libieee

# Architecture: changes for Tcl 8.6

- Obsoleted by Tcl 8.5:
  - rechan: reflected channels now part of core
  - pwb: encoding bugs fixed
- Obsoleted by Tcl 8.6:
  - zlib: now included in core
- Obsoleted by glibc 2.x
  - libieee

---

Recent TclVFS (1.4.2+) looks for new built-in funcs before fallback to old rechan and zlib APIs. Thus a basekit with upgraded TclVFS and Tcl 8.6 can exclude these old libs entirely!

# Pursuit: modularity, configurability

- Traditionally, basekits have been built by single scripts, script/Makefile combos:
  - Limited configurability
  - Dense
  - Intimidating to hack, debug
  - Barrier to upgrading, expanding, custom config.

# Pursuit: modularity, configurability

***New Tools:***

- TEPAM
  - "Tcl's Enhanced Procedure and Argument Manager"
    - Part of Tcllib
    - Argument parser
      - Full-featured (customizable constraints, error handling)
      - Self-documenting
    - Can define sub-commands (like an ensemble), and sub-sub-commands

```
$ ./buildkit -help
NAME
      buildkit -
      Build a bare-bones Tclkit.
SYNOPSIS
      buildkit
            [-tcl_version <tcl_version>]
               Tcl version to use in Tclkit, default: "8.6.8"
            [-tk_version <tk_version>]
               Tk version to use in Windows Tclkit, default: "8.6.8"
            [-metakit_version <metakit_version>]
               Metakit version to use in Tclkit, default: "2.4.9.7"
            [-vfs_version <vfs_version>]
               TclVFS version to use in Tclkit, default: "1.4.2"
            [-target <target>]
               Target to pass to make program for building, default: "tclkit"
            [-compress <compress>]
               Specify if files in tclkit vfs are to be stored in compressed
               form.  Value of '1' is compressed, '0' is uncompressed, type:
               integer, default: 1
            [-platform <platform>]
               OS platform to build for. currently supported: unix, windows,
               default: "unix"
DESCRIPTION
       Builds a Tclkit with only Tcl, TclVFS and Metakit shared libraries. Only
       pure-Tcl packages and a few encoding files are included in the TclVFS.
       ---
       examples:
       -
       buildkit -target distclean   # deletes all compiled objects and generated
       Makefiles.
       -
       buildkit -tcl_version 8.6.8   # builds tclkit using Tcl ver. 8.6.8.
       -
       buildkit   # builds tclkit with all default values.
```

# Pursuit: modularity, configurability

***New tools:***

- tmake.tcl
  - Pure-Tcl partial make clone
  - Part of Ghostscript project
  - Affero GPL
  - tmake file is a valid makefile
  - Accepts command line var settings
  - Can include simple makefile, extract vars, rules
    - Optionally customize make environment per platform, etc.

# TEPAM + tmake.tcl → basekit

- TEPAM-driven master script “buildkit”:
  - Define high-level command-line options, targets
  - Write make target recipe code as TEPAM subcommand
  - Format make vars and pass them to tmake command line...
- tmake makefile:
  - Isolate each build component into separate target
  - Target rule calls buildkit subcommand to execute recipe
    - *(can write build steps as a Tcl script rather than shell script calls)*
  - Configure, customize, validate, debug, upgrade each target separately

# Results:

- Documented, transparent, modular, portable, auditable, hackable build project
- Two files:
  - buildkit (5 kB)
  - makefile (4 kB)
- Future-ready
- Obsolete code eliminated
- Upgrades:
  - Tcl → 8.6.8
  - TclVFS → 1.4.2
  - Metakit → 2.4.9.8

# Future Work:

- SQLite – backed basekit:
  - A SQLite VFS shim file has recently been committed to the SQLite fossil repository that allows a SQLite database to be appended onto the end of another file, such as an executable.

# Future Work:

- Make files/shared libs in Starpack visible to operating system:
  - Unix: Incorporate pure-Tcl FUSE read-only client
    - (http://wiki.tcl.tk/13853)
  - Mount FUSE client at program start
  - Windows: Incorporate pure-Tcl FTP server from Tcllib
  - At program start, boot FTP server and create filesystem letter drive fed by it

# Future Work:

- Replace Autoconfig tools in basekit kitsh sub-project with pure-Tcl Autosetup
  - https://msteveb.github.io/autosetup/

---

"autosetup is a tool, similar to autoconf, to configure a build system for the appropriate environment, according to the system capabilities and the user-selected options."

# *A Reboot of the Starpack Build Process*

## *For Tcl 8.6 and beyond*

Steve Huntley
Health Information Technology Lab
University of Maryland, Baltimore County

Tcl/Tk Conference 2018