Nathan Coulter

Curriculum Vitae

Hang the sense of it all and keep yourself occupied

Education

Bachelor of Science, Liberal Arts, *Excelsior College*, Albany, NY

Professional Experience

Software Engineer, *KomTere OÜ*, Tallinn, Estonia

- Providing technical direction and expertise for all business activities.

Software Engineer, Systems and Database Administrator, *Metaladder*, Salt Lake City, UT

- Foresaw the need to create and maintain software collections across an array of platforms.
  - Built such a system and used it to deploy and manage over 1500 software distributions into a cohesive collection used in a mission-critical FDA-audited workflow.

Systems and Database Administrator, *Intermountain Injury Control Research Center*, Salt Lake City Utah

Managed databases and applications for statisticians and researchers

Computer Languages

Tcl
Advanced.

Python
Advanced.

Unix Shell
Advanced.

C, Java
Intermediate.

SQL
Advanced.

R
Intermediate.

Rust
one year of experience

Javascript, Lisp, Haskell
Conversant.

1006 W. 1100 N. – Pleasant Grove, UT 84062 – USA
+1 (801) 604 3341 • *+1 (801) 604 3341*
curriculumvitae2021@pooryorick.com • *www.pooryorick.com*
pooryorick

IT Fields of Expertise

General programming
proficient

Windows systems administration
proficient

Unix systems administration
proficient

database administration
proficient

Specific Technologies

Experience with the following:

other languages	Bash, Component Pascal, LaTeX
databases	MySQL, PostgreSQL, SQLite
filesystems	AFS, NFS, EXT4
CMS	Documentum
Tools	GDB, Valgrind, Nix/NixOS

Spoken Languages

English	Native.
Russian	Proficient.
Estonian	Proficient.
Chinese	Proficient.
German	Intermediate
Spanish	Intermediate

1006 W. 1100 N. – Pleasant Grove, UT 84062 – USA
+1 (801) 604 3341 • +1 (801) 604 3341
curriculumvitae2021@pooryorick.com • www.pooryorick.com
pooryorick

Areas of Expertise

Nathan Coulter

May 15, 2025

Contents

Chapter 1

Technical Works

1.1 Rewrite of a email message processor

The venerable mime package in Tcllib was written in 1991 by Marshall T. Rose, who together with Nathaniel Borenstein wrote Safe Tcl, which was later merged into the Tcl Core as part of the interp command. In 2018 I redesigned the mime package in order to expand its support for current standards.

1.1.1 required expertise

- RFC 2045
- RFC 2046
- RFC 2047
- RFC 3875
- RFC 2231
- RFC 6265

1.1.2 work performed

- For cookies, add HttpOnly parameter and make it the default.
- Remove unnecessary semicolon in a cookie header that contains no parameters. For a header value, add parameters are simple values rather than key/value pairs.
- mime: remove "-" from cookie options.
- Add [cookie delete]
- mime and ncgi: update documentation, cleanup variable names, call methods through their instances rather than directly.
- Tigthen up datetime scanning.

- Use a command instead of a token to represent a mime message. Improve error propagation in testutilities.
- Remove the "token" argument of [::mime::header::serialize].
- Internal rewrite plus API redesign. ncgi: Use mime module instea of bespoke routines
- [header get] now returns one matching value, plus parameters. Documentation updated to reflect new API.
- ncgi:ncgi: Separate form and query parsing. New [post], [poststring] and [querystring] commands. Changed [query] command.
- Replace [buildmessage] and [copymessage] with [serialize]. Replace [getbody] with [body] and eliminate the -command option.
- mime: Added -chan option. More thorough cleanup of tests. Refreshed documentation.
- TCL fileutil::magic::mimetype not recognising Microsoft documents or mp3
- Remove duplicated code in ::mime::buildmessageaux
- Numerous improvements and bug fixes. Updated analyzer script from file-5.34 magic files.
- Internal rewrite plus API redesign. ncgi: Use mime module instea of bespoke routines. New modules: ego chan
- Use a command instead of a token to represent a mime message. Improve error propagation in testutilities.
- Remove unnecessary semicolon in a cookie header that contains no parameters. For a header value, add parameters are simple values rather than key/value pairs.
- For cookies, add HttpOnly parameter and make it the default.
- [header get] now returns one matching value, plus parameters. Documentation updated to reflect the new API.
- ncgi: use the mime module to generate the response headers mime add -addmimeversion and -addcontentid options
- Fix problem with content type handling. ncgi: Add object interface, update documentation.
- Add RFC 2231 support.
- Let ::mime::parsepart accept a header only with multipart. Headers and body are parsed separately, and only as needed.
- mime::initialize - make multipart/mixed the default type when -parts is provided.

- Error: too much braced message header values (::mime::getheader). Fixed by allowing multiple redundant keys in header records.
- Problems with qp_encode and encoded_word, no_softbreak
- Make it a no-op instead of an error to set a "read-only" header to an identical value, and allow deletion of a "read-only" header.
- Separate header determination from body generation.
- If nothing else, at least indicate whether a file is text or binary
- Correct Handling of quotes in quoted strings in parseMimeValue

1.2 ycl shelf

A prototype-based object system written in Tcl. It is perhaps the only fully-featured prototype-based system built on TclOO.

1.3 expertise required

- object orientation
- Tcl
- TclOO,

1.4 work performed

- ycl shelf
- test suite

1.5 Redesign Cleanup Routines of an Object System Written in C

When I rewrote ycl shelf to use TclOO, I found that the code paths I was taking were rather uncharted, and various changes to TclOO were needed. In the process of making these changes, I found it necessary to untangle the web of interactions between routines responsible for cleaning up objects and classes upon deletion. I iteratively refactored the routines until they became much more straightforward. It was painstackingly detail-oriented work.

1.5.1 required expertise

- C
- TclOO
- Tip 257: Object Orientation for Tcl
- TIP 279: Adding an Extensible Object System to the Core

1.5.2 work performed

- Fixed memory leak in TclOO.c:ObjectNamespaceDeleted, object mixins and object/class mutation
- Fix more memory leaks in TclOO cleanup routines
- Segmentation fault: deleting a class mixed into an object that calls a method on the class in a deletion trace
- Further refactoring of TclOO object cleanup routines
- TclOO segmentation fault cleaning up objects that that have mixed themselves into themselves
- Streamline TclOO object cleanup routines
- Segmentation fault in TclOO.c/ReleaseClassContents() for a class mixed into one of its instances
- Aborts when a trace on command deletion deletes the object's namespace
- Method with non-literal value for body argument causes segmentation fault

1.6 Redesigned cleanup routines of Tcl namespace system

After completing the work to simplify TclOO, I realized that the existing implementation of Tcl namespaces had similar issues, and that similar improvements could be made to it. I started this work immediately in order to rely on recently-gained understanding before it faded from memory.

Namespaces are a fundamental component of Tcl. A namespace is a container for commands, variables, and other namespaces. When a namespace is deleted, all relevant resources must be cleaned up in a way that doesn't run afoul of possibly complex dependencies between those resources. The existing cleanup routines had become more complex over time as patches accredited to fix reported bugs. I rewrote these routines, significantly simplifying them in the process. It was tricky work because it required in-depth knowledge of all the moving parts inside Tcl interpreter.

1.6.1 required expertise

- C
- Tcl

1.6.2 work performed

- Tcl Tip 498: Simplify Tcl_DeleteNamespace
- Resources of a namespace are still available after the namespace is deleted.

- Namespace deletion: trace that calls [apply], specifying the namespace, doesn't fire.
- Delete a namespace for an ensemble having a deletion trace deletes its namespace: segmentation fault.

1.7 A System for Building and Deploying Software across a Heterogeneous HPC Computer Network

Challenges of making software available in an HPC environment include providing multiple versions and configurations of the same software, ensuring that programs and libraries are able to load the correct dependencies, supporting conflicting versions of the same libraries, managing local software patches, ensuring a reproducible build-time environment, and supporting subordinate package management systems such as Python, R, and Perl. I wrote and maintained this system for over a decade for a large client. It deploys over 2000 packages to software collections that are isolated from the collections provided with the OS distribution. The build system ensures that each compiled program and library searches in an isolated location for dependencies, allowing fine-grained control over software lifecycles and updates. Additionally each compiled program and library is constructed such that it can be migrated from development to production areas by external tools.

1.7.1 required expertise

- Linkers, Loaders, and Libararies

1.7.2 work performed

- implementation

1.8 NRE-enabled [package require]

[package require] was not previously NRE-enabled, as the C implementation evaluated scripts in multiple loops. This complexity made NRE-enabling a somewhat daunting challenge. It is by far the most complex command to have been coroutine-enabled.

1.8.1 required expertise

- C
- Tcl TIP 328: Coroutines

1.8.2 work performed

- implementation

1.9 A hybrid scoped programming system and hierarchical data structure

This data structure acts both to provide scoping in programming language constructs and to provide a hierarchical data structure that can be navigated using the provided API, which incluse XPath queries.

1.9.1 required expertise

- Document Object Model
- Tcl
- Tdom

1.9.2 about

- EuroTcl 2018: Ycl Env

1.9.3 work performed

- implementation
- test suite
- selectNodes -cache 1 and a syntactically incorrect xPath statement

1.10 A Novel Implementatin of the File command

This implementation, originally by Colin McCormack, consumes magic files provided by The Fine Free File Command and "compiles" them into a performant Tcl script.

1.10.1 required expertise

- The Fine Free File Command

1.10.2 work performed

- deal with comments on a mimetype line
- workaround for bad syntax in Magdir/dos
- correct processing of operations on indirect offsets. Correct handling of "default" tests. Fix bug in level handling.
- Fix bug in whitespace processing of Smatch command
- fumagic Add support for ustring
- module fumagic eliminate [procdef], replace with [generate]

- significant changes to fumagic to bring it closer to feature-parity with file(1). Changed format of [filetype] result.
- ensure in ::filetypes::magic::filetype that %c substitutions happen.

1.11 Daerth: A Distributed Pipeline for Multiprocessing

In this pipeline workflow, a sequence of queues track outstanding work to be performed, and the state of that work. Each worker is an atonomous agent that retrieves a task from one queue, performs the requisite work, and places the results on the next queue. A worker can independently choose a queue to work on, workers can be widely distributed across the network, and work for hire can be advertised to a broad audience. This design removes bottlenecks, making it possible to set up massively distributed and scalable processing operations.

In the process of developing this work, a deficiency in the Thread Package for Tcl was discovered. This deficiency caused brought the Thread package to a halt in various easily-reachable scenarios, and was fixed in the course of completing this work.

1.11.1 read more

- Daerth

1.11.2 required expertise

- C
- Tcl
- Thread Package for Tcl

1.11.3 work performed

- implementation
- test suite
- -eventmark + thread::send -async == deadlock

1.12 Tcllib Infrastructure Improvements

1.12.1 work performed

- devtools: Make it possible to use min-max version specification with testsNeedTcltest.
- New module: chan coroutine: specializes channel commands to use the current coroutine for non-blocking I/O management.
- Make chan getslimit a routine that specializes an existing channel command rather than a channel command itself.

- Fix issues: ticket d74e418e84, [coroutine::util read] closes channel, and 104809e450, [coroutine::util read] swallows data.
- sak: To improve test isolation, give each .test file its own interpreter.
- devtools: Fix bug where a message was not produced because a substitution failed.

1.13 Tcllib: Miscellaneous Improvements

1.13.1 required expertise

Tcl Uniform Resource Identifier (URI): Generic Syntax

1.13.2 work performed

- Improve error propagation in fileutil::foreachLine.
- Fix namespace resolution for relative namespaces. Add "normalize" and "strip" commands.
- Modification to ::uri::GetUPHP to make URI parsing a little more conformant.

1.14 Improving Tcl

Tcl is composed of a rich set of subsystems, and fixing an issue that is discovered often requires in-depth knowledge of several subsystems.

1.14.1 required expertise

- C
- gdb
- Tcl
- TclOO,

1.14.2 work performed

- ensemble: segmentation fault when -subcommand and -map values are the same object
- Incorrect error reporting in traces
- [info exists array(elem)] deletes array searches
- EnvTraceProc errors when env var doesn't exist

1.15 Improving TclOO

1.15.1 required expertise

- C
- TclOO

1.15.2 work performed

- TclOO segmentation fault cleaning up objects that that have mixed themselves into themselves.
- namespace ensemble command named ":" is mistakenly given the empty string as its name
- Tcl_Eval() causes new segmentation fault
- TclOO - mixing an object results in segmentation fault
- TclOO method with non-literal value for body argument causes segmentation fault

1.16 Finding Memory Errors in Programs Written in C

Errors of this category can be difficult to track down. It's an interdisciplinary activity that requires a thorough understanding of the machine, the language, the platform, and the available degugging tools.

1.16.1 required expertise

- C
- gdb
- Tcl
- TclOO,

1.16.2 work performed

- epoll notifer leaks memory

1.17 Ego: A Minimal But Complete Object System

I wrote Ego after studying numerous object systems for over a decade. Ego is my submission to illustrate that a fully operational prototype-based object system can be extremly concise.

1.17.1 required expertise

Tcl

1.17.2 work performed

- implementation
- test suite

1.18 A System Similar to async/await for Communication between Coroutines

Coroutines are often used in Tcl in conjunction with the event loop to provide something like Green Threads. Using this system, a coroutine can engage in a conversation with another coroutine using normal call semantics, along with a return code of "break" to indicate the end of the conversation.

1.18.1 required expertise

- Tcl

1.18.2 work performed

- implementation
- test suite

1.19 A Batteries-Included Tcl distribution for Cygwin

To build and install a reasonably complete set of packages for Tcl, it was necessary to slightly modify the Cygwin package management system itself, and then to create a port of each included Tcl extension.

1.19.1 required expertise

- Tcl
- Cygwin
- autoconf
- GNU Make
- m4

1.19.2 work performed

- Tcl autoconf patch
- implementation

Chapter 2

Presentations

2.1 EuroTcl 2018: Ycl Env

url https://www.youtube.com/watch?v=BQbMuWB9oNM

2.2 Euro Tcl 2017: An Event Loop For Coroutines

url https://www.youtube.com/watch?v=WGYl0LiPv80

2.3 EuroTcl2 2016: Coroutine is the new Main

url https://www.youtube.com/watch?v=yCC--STRzo4

Chapter 3

Technical Documentation

Learn X in Y Minutes: Tcl Covers all aspects of the Tcl language, concisely, and by example.

Chapter 4

Bug Reports

An important aspect of developing software is the art of diagnosing and reporting bugs in such a way that the maintainers of a piece of software have a case the concisely and informatively illustrates the errant behaviour. The following is a list of issues that I have diagnosed and reported to the relevant maintainers.

- [dict exists] doesn't produce an error for non-dict values.
- trace generates of string reprepresentatin of "commandPrefix" argument.
- vwait not NRE-enabled: C stack busy.
- A trace deletes the namespace when a command is replaced, producing "called Tcl_CreateHashEntry on deleted table"
- TclOO: segmentation fault: deleting a class mixed into an object that calls a method on the class in a deletion trace.
- exec redirection to append stdout and stderr to the same file.
- [dict with] discards string rep even when it doesn't modify the dictionary.
- memory cost and execution time exploding of nested namespaces.
- glob, -nocomplain, -types, and insufficient privileges.
- [glob], [encoding system] and encoding-free filesystems.
- interp alias to command whose name is the empty string.
- refchan, coroutine, and postevent from the "watch" proc.
- rename ::set and lose custom return options.
- lassign + dict update == segmentation fault.
- [uplevel *list*], [*uplevel*1list] and generation of string representations.
- a namespace loses track of itself.
- interpreter initialization, auto_path, [info nameofexecutable], and symbolic links

- platform::identify: regexp doesn't match platform.
- regexp -indices documentation doesn't mention matchVar.
- in a thread, switch a non-blocking channel to blocking and close == hang.
- tclsh in a command pipeline: ::tcl_interactive is 0, but interpreter is in interactive mode (halfway).
- fCmd-27.5 and 27.6 fail because permission to set attribute is denied.
- alias trace did not find command in real command list of import references.
- granularity other than 1 allows interp to exceed limits.
- copied object member variable confusion.
- platformCPUID segmentation fault on Darwin.
- INST_STORE, TclObjLookupVarEx, createPart*.
- [array set] inside [eval] inside proc: bad stack adjustment.
- trace that unsets and then sets and array.
- expr mathfunc comma separator for arguments.
- ranlib after install with read-only permissions.
- off-by-one address error in AppendUnicodeToUnicodeRep.

Chapter 5

Teaching and Mentoring

5.1 Internship

5.1.1 Internship review

In the spring and summer of 2018, Mr. Coulter ran an internship program for Komtere OÜ. Three interns received certificates for completing 300 hours of the internship program. One student went on to receive a second certificate for completeing 360 hours, and later became an employee of Komtere OÜ.

During the internship, students were introduced to principles of Unix computing. Topics included the Filesystem Hierarchy Standard, hard and soft links, built-in shell commands, standard channels, and core utilities. Interns set up a Unix software development environment and over the course of the internship developed a framework for publishing web applications. They used Fossil to organize and manage source code, and althttpd, to interface via CGI with Tcl system that generated HTML documents. As part of the training, interns turned to the relevant specifications and reference manuals rather than using Internet Search Engines. The system included a registration, login, and session management functionality.

Interns also learned how to use the the gcc toolchain to compile and link programs and libraries.

Annabel Tsarjov described her experience leading up to the internship, the internship itself, and her subsequent activities:

> I did not have any experience in computer science nor programming, before I joined the internship programme held by Komtere OÜ. I was previously inspired by Nathan to start learning programming and wanted to get some experience before I started my studies in university. I took the offer to the internship programme with the intention of learning as much as possible and be the best candidate in order to be employed in the future.
>
> I really enjoyed the whole internship process because I found it very in depth. I learned everything in great detail, which might have seemed a little annoying during the internship but granted a lot more knowledge of everything I learned there and reshaped the way I study. At first, I was little intimidated by the fact that the other

interns had previous experience with programming and therefore, would be grasping the information quicker then I did, but quite soon I was able to get to the same level as they were and even surpass them.

After the internship I started my university studies at Tallinn University of Technology and have successfully finished my first semester. From this February I was offered a job in Komtere OÜ as an IT specialist.

5.2 Tcl conferences

Tcl conferences are held in the US and in Europe each year and they are meant to present the changes in the language and on going projects using the Tcl programming language.

5.2.1 EuroTcl 2016: Coroutine is the new main

- The link to the video recording of the conference.

The presentation is about what coroutines are, how they work and what they are. The presentation starts with simple examples and goes on with introducing a self created toolkit for coroutines.

5.2.2 EuroTcl 2017: An Event Loop For Coroutines

- The link to the slides of the presentation.
- The link to the video recording of the conference.

5.2.3 EuroTcl 2018: {ycl env} and Clone Don Porter project

- The link to the video recording of the presentation.
- The link to Clone Don Porter project

The presentation starts of with an project called Clone Don Porter project which was about fixing issues in the Tcl language.

Chapter 6

Conferences Attended

Chapter 7

Subject Matter

7.1 autoconf

& https://www.gnu.org/software/autoconf/

7.2 The C Programming Language

url http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf

7.3 The Common Gateway Interface (CGI) Version 1.1

url https://tools.ietf.org/html/rfc3875

7.4 Cygwin

url https://www.cygwin.com/

7.4.1 Cygwin package management system

url https://github.com/cygwinports

7.5 gcc

url https://gcc.gnu.org/

7.6 gdb

url https://www.gnu.org/software/gdb/

7.7 Document Object Model

7.7.1 Document Object Model (DOM) Level 1 Specification

url https://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/

7.7.2 Document Object Model (DOM) Level 2 Core Specification

url https://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/

7.7.3 Document Object Model (DOM) Level 3 Core Specification

url https://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/

7.8 The Fine Free File Command

url http://darwinsys.com/file/

7.9 Fossil

url https://fossil-scm.org/index.html/doc/trunk/www/index.wiki

7.10 RFC 6265: HTTP State Management Mechanism

url https://tools.ietf.org/html/rfc6265

7.11 GNU Make

url https://www.gnu.org/software/make/

7.12 m4

url https://www.gnu.org/software/m4/

7.13 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

url https://tools.ietf.org/html/rfc2045

7.14 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

url https://tools.ietf.org/html/rfc2046

7.15 MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text

url https://tools.ietf.org/html/rfc2047

7.16 MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations

url https://tools.ietf.org/html/rfc2231

7.17 Internet Message Format

url https://tools.ietf.org/html/rfc5322

7.18 Linkers, Loaders, and Libraries

7.18.1 How to Write Shared Libraries

url https://www.cs.dartmouth.edu/sergey/cs258/ABI/UlrichDrepper-How-To-Write-Shared-L
author Ulrich Drepper

7.18.2 Linker and Libraries Guide

url https://docs.oracle.com/cd/E19683-01/817-3677/817-3677.pdf

7.19 object orientation

7.19.1 reference

- Classifying Prototype-based Programming Languages, C. Dony , J. Malenfant and D. Bardou
- Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems , Henry Lieberman, Artificial Intelligence Laboratory, Massachussetts Institute of Technology

7.19.2 The Open Group Base Specifications Issue 7, 2018 edition

url http://pubs.opengroup.org/onlinepubs/9699919799/utilities/file.html

7.20 Tcl

url https://en.wikipedia.org/wiki/Tcl

7.21 Non-Recursive Engine for Tcl

url https://wiki.tcl-lang.org/page/NRE

7.22 TclOO

url https://wiki.tcl-lang.org/page/TclOO

7.23 Tcl TIP 257: Object Orientation for Tcl

url https://core.tcl.tk/tips/doc/trunk/tip/257.md

7.24 Tcl TIP 279: Adding an Extensible Object System to the Core

url https://core.tcl.tk/tips/doc/trunk/tip/279.md

7.25 Tcl Tip 328: Coroutines

url https://core.tcl.tk/tips/doc/trunk/tip/328.md

7.26 Tdom

Tdom is an implementation of the Document Object Model.

url https://tdom.org/index.html/artifact?filename=doc/dom.html

7.27 Thread Package for Tcl

url https://www.tcl.tk/man/tcl/ThreadCmd/thread.htm

7.28 Uniform Resource Identifier (URI): Generic Syntax

7.28.1 RFC 3986, 2005

url https://tools.ietf.org/html/rfc3986

7.28.2 RFC 1738, 1994

url https://tools.ietf.org/html/rfc3986

7.29 XPath

7.29.1 XML Path Language (XPath) 3.1

url https://www.w3.org/TR/xpath-31/