A CURMUDGEON'S LANGUAGE SELECTION CRITERIA:

WHY I DON'T WRITE EVERYTHING IN GO, RUST, ELIXIR, ETC.

G. CLIFFORD WILLIAMS

GCW@8IONS.COM

CONSULTANCY: INFRASTRUCTURE AUTOMATION

- Unix-y systems engineering (*BSD, Solaris, etc)
- Private Cloud Buildout (OpenStack, OpenShift, Triton, etc.)
- CI/CD Pipelines
- Tools development (Scripts, Web UIs, etc)
- Systems Integration
- Multi-platform tooling

ABOUT ME

- Politics: BSD > GNU
- Religion: Null
- Languages: Shell, Awk, Python, Racket, Lua, C, Rust, Go, Elixir, Crystal, Dart, JavaScript, Julia
- Not an expert in any of the above
- No Degree

WHAT THIS TALK IS NOT

- Tutorial
- Trashing any language
- Pushing any language
- Terribly Intellectual and Abstract
- Religious

THINGS THIS TALK IS NOT

TUTORIAL

- If you want to learn these languages go learn them:
  - Go – https://tour.golang.org/welcome/1
  - Rust – https://doc.rust-lang.org/book
  - Elixir – https://elixir-lang.org/getting-started
  - Crystal – https://crystal-lang.org/docs/
  - Julia – https://julialang.org/learning/

COMPREHENSIVE

- Too many languages
  - Dart – https://www.dartlang.org/
  - Nim – https://nim-lang.org/
  - Swift – https://docs.swift.org/swift-book/
  - Kotlin – https://kotlinlang.org/

COMPREHENSIVE

- Too many domains
  - Web development
  - Integration Tooling (API clients and endpoints)
  - Systems Tools/Utilities (batch processing, maintenance, administration, etc)
  - Embedded
  - Automation & Orchestration

UNBIASED

- I'm human
- I lean toward "…but did you ship?"
- I lean toward "default tools" (Shell, Awk, C, etc)
- My language preferences:
  - Small mental footprint (number of keywords, syntax, core statements/ functions, ceremony)
  - Low cognitive overhead (type considerations, features, idioms, syntax, batteries [included or sold separately], community sanity)
  - Standards (POSIX, libc, etc.)

TRASH TALK OR PROMOTION

- I have no desire to turn you away from any language
- I have no desire to push you toward any language

TERRIBLY INTELLECTUAL OR ABSTRACT

- Efficiency of data structures
- O notation
- Reflection vs introspection
- Message passing vs method calling

RELIGIOUS: SCREW YOUR PURITY

- Functional
- Inheritance / Composition
- Iterative / recursive
- Concurrent / parallel / async
- Polymorphism
- *We **need** purists in the community. We do **not** need a community of purists.*

LANGUAGE LANDSCAPE

PRE-LANGUAGE HISTORY

- Hero(n) of Greece (programmable puppet machine)
- Abaci (bead calculators)
- Programmable Looms
- Babbage machine (shout out to Ada Lovelace)
- “modern” languages and compilers

LANGUAGES

- High Level vs Low Level (asm vs C, C vs Python, etc)
- Dynamic vs Static
- Type systems
- Compiled vs Interpreted

FOCUS

- System tools/utilities
- API clients and endpoints
- Web Apps (even web-to-desktop)
- Not:
  - Mobile apps
  - Embedded

PERFORMANCE VS CONVENIENCE

- Low-level == Fast(er?)
- Highlevel == Easier

ASSEMBLY LANGUAGE

- Architecture dependent
- Low overhead

C / C++ / OBJ-C

- High(er) Level (than ASM)
- Risky
  - signed/unsigned integers
  - pointer math
  - memory management

PYTHON / RUBY / JAVASCRIPT

- Super High-level
- Powerful tools for beginners
- Garbage collected (no memory errors???)
- Hooks for integrating with lower level code (FFI and others)

LANGUAGES THAT AREN'T C/C++ AND AREN'T PYTHON/RUBY

- Lisp / Scheme / Racket
- Lua
- Erlang
- Haskell
- Ada / Spark

PANACEA

- High performance (everyone wants C's speed without C's insecurity)
- Super high-level feeling (ease of Python/Ruby)

GO

- Focus on concurrency and code adaptability
- Obsessive about feature justification
- Short compile times
- Very small language
- Good performance characteristics
- Self hosting and re-invented the wheel…every wheel

RUST

- Memory Lifetimes (instead of garbage collection)
- Lending / Borrowing (safety?)
- Concurrency
- Powerful testing features
- Great packaging and ‘vendoring’ management
- Highly performant

ELIXIR

- Erlang's power with Ruby-like syntax
- Highly concurrent
- Super functional (purity?)

WHY NOT GO?

- Overselling memory safety
- Go runtime and garbage collector
- Go – libc and linux bias
- Cognitive overhead – array “slices”
- Platform support
  - Did I mention linux bias?

WHY NOT RUST?

- Overselling memory safety
- Linux bias
- Cognitive overhead – memory safety scheme
- Platform support
  - Did I mention linux bias?

WHY NOT ELIXIR?

- Cognitive overhead – Elixir Syntax can be odd in places
- Debugging in Erlang
- Deployment/packaging issues

SYSTEM TOOLS/UTILITIES

- Maintenance over everything else
- System requirements (runtimes, libraries, packaging, etc)
- My language order:
  - Shell (POSIX… not bash) or AWK
  - Lua
  - Python
  - C
  - Honorable mention: Nim (go check it out)

WEB DEVELOPMENT

- Frameworks (leveraging other people's work)
- Deployment considerations (packaging, automation, etc)
- Developer availability
- My language order:
  - Python
  - Go
  - Elixir
  - Lua
  - Honorable mention: Racket (go learn about it)

SYSTEMS INTEGRATION (API WORK)

- Pre-existing libraries (leveraging other people's work)
- Platform support
- Maintainability
- My language order:
  - Python
  - Lua
  - Go

LANGUAGES I LIKE

- Ada / Spark
- Racket
- Shell (posix)
- Ksh (93+)
- AWK
- NIM
- Lua

RANKED LIST

- Portability
- Maintainability
- Cognitive overhead (special case for things I work with every day – that's hypocrisy)
- Code leveraging
- Fiddly bits (macros, meta-programming, tooling, etc)

THANK YOU

GCW@8IONS.COM