A Tour of New Systems Languages

Russel Arbore

Obligatory s***ing on C++

- The standard is not open - technically, you’re supposed to purchase it

Obligatory s***ing on C++

- The standard is not open - technically, you’re supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf

Obligatory s***ing on C++

- The standard is not open - technically, you’re supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf
- The standard is 1448 pages long

Obligatory s***ing on C++

- The standard is not open - technically, you’re supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf
- The standard is 1448 pages long
- Too many language features

Obligatory s***ing on C++

- The standard is not open - technically, you’re supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf
- The standard is 1448 pages long
- Too many language features
  - “friend”?????

Obligatory s***ing on C++

- The standard is not open - technically, you're supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf
- The standard is 1448 pages long
- Too many language features
  - "friend"?????
- *Super* unsafe

Obligatory s***ing on C++

- The standard is not open - technically, you're supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf
- The standard is 1448 pages long
- Too many language features
  - "friend"?????
- *Super* unsafe
- Not typesafe

Obligatory s***ing on C++

- The standard is not open - technically, you're supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf
- The standard is 1448 pages long
- Too many language features
  - "friend"?????
- *Super* unsafe
- Not typesafe
- *Very* slow compile times

Obligatory s***ing on C++

- The standard is not open - technically, you're supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf
- The standard is 1448 pages long
- Too many language features
  - "friend"?????
- *Super* unsafe
- Not typesafe
- *Very* slow compile times
- Bad tooling

Obligatory s***ing on C++

- The standard is not open - technically, you're supposed to purchase it
  - https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf
- The standard is 1448 pages long
- Too many language features
  - "friend"?????
- *Super* unsafe
- Not typesafe
- *Very* slow compile times
- Bad tooling

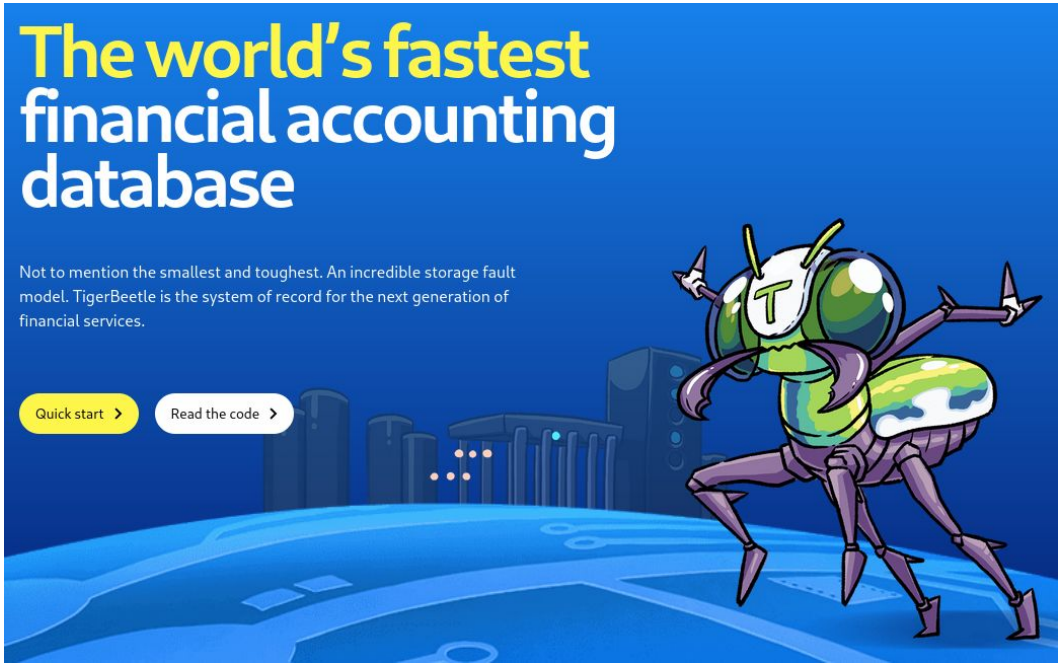
Every other language talked about improves on some of these components.

- Started in 2006
- Introduced the borrow checker
- Type safe
- Many functional patterns
- Cargo

- Started in 2006
- Introduced the borrow checker
- Type safe
- Many functional patterns
- Cargo
- Slow compile times
- “Hard to program”
- Types can get gnarly
  - Arc<Mutex<...>> spam
- *Not* simple
- Inspired many other new systems languages

- Started in 2016
- Simple
  - Control flow / memory allocations are always explicit
- Comptime
- Works alongside C/C++
  - zig build (libclang to compile C/C++)
- Friendly to embedded / freestanding environments

- Started in 2016
- Simple
  - Control flow / memory allocations are always explicit
- Comptime
- Works alongside C/C++
  - zig build (libclang to compile C/C++)
- Friendly to embedded / freestanding environments
- Same memory guarantees as C (🤣)
  - -Debug and ReleaseSafe help
- If Rust is a C++ replacement, Zig is a C replacement
- Development seems very pragmatic


The world's fastest financial accounting database
Not to mention the smallest and toughest. An incredible storage fault model. TigerBeetle is the system of record for the next generation of financial services.
Quick start
Read the code

Performance

Orders of magnitude more performance with room to spare.

Faster than a generic in-memory database but with replicated persistence for every transaction, zero deserialization with cache line aligned data structures, zero copy with Direct I/O, zero syscalls with io_uring, and static allocation of memory (and storage). More performance reduces cost and leaves a large margin of safety to absorb the unexpected. TigerBeetle is ludicrously fast with a small footprint to boot. Why big iron when you can beetle?

1000x
faster than ad hoc balance tracking

50%
more efficient than a one-phase ledger

50%
more write availability in the critical path

20%
smaller clusters with flexible quorums

Attempting to summarize,

- Rust is about compositional safety, it's a more scalable language than Scala.
- Zig is about perfection. It is a very sharp, dangerous, but, ultimately, more flexible tool.

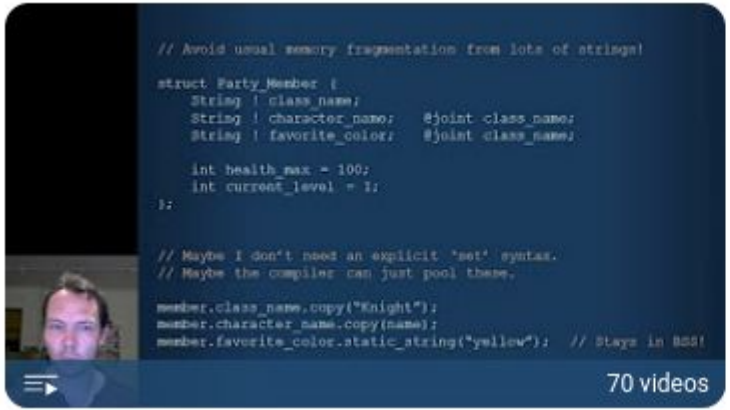
https://matklad.github.io/2023/03/26/zig-and-rust.html

https://www.scattered-thoughts.net/writing/assorted-thoughts-on-zig-and-rust/

- Started in 2014
- Created by Jonathan Blow
- Lots of control of memory layout
- Powerful metaprogramming system
  - Macros can operate on Jai AST
- Fast compile times (~100k-200k LOC/s)
- Simple build system
  - jai main.jai
- Standard libraries for graphics / audio / assets

- Started in 2014
- Created by Jonathan Blow
- Lots of control of memory layout
- Powerful metaprogramming system
  - Macros can operate on Jai AST
- Fast compile times (~100k-200k LOC/s)
- Simple build system
  - jai main.jai
- Standard libraries for graphics / audio / assets
- Compiler isn’t open
- Same memory guarantees as C (🤣)
- Geared towards performance-critical application development

https://www.youtube.com/watch?v=TH9VCN6UkyQ&list=PLmV5I2fxaiCKfxMBrNsU1kgKJXD3PkyxO



A Programming Language for Games

Jonathan Blow

Ideas about a new programming language for games. • 1:55:24

A Programming Language for Games, talk #2 • 1:30:26

VIEW FULL PLAYLIST

- Started in 2016
- Created out of frustration with C++
- Simple and minimal
  - Orthogonality - one way to write something
  - Small language standard
- Data oriented
- Defer
- Many pragmatic choices
  - Built-in dynamic array, UTF-8 string, map, context, allocators

- Started in 2016
- Created out of frustration with C++
- Simple and minimal
  - Orthogonality - one way to write something
  - Small language standard
- Data oriented
- Defer
- Many pragmatic choices
  - Built-in dynamic array, UTF-8 string, map, context, allocators
- Even more minimal than Zig
- Same memory guarantees as C (🤣)

Odin in Production

JangaFX are the creators of the 3D animation software EmberGen written *fully* in Odin.

EmberGen is a real-time volumetric fluid simulator that can instantly simulate, render, and export flipbooks, image sequences, and VDB volumes. With EmberGen, you can create anything from fire and smoke, to explosions and magic wisps. EmberGen gives you the creative freedom to iterate on your simulations in a few milliseconds instead of hours.

Through EmberGen, Odin runs in production among the giants of the games and film industries: Bethesda, CAPCOM, Codemasters, THQNordic, Warner Bros, Weta Digital, and many others.

- Open sourced in 2019
- No undefined behavior
- Sum types
- Zero dependency binaries
- Can translate C to V automatically
- Fast compile times (as high as 500k LOC/s)

- Open sourced in 2019
- No undefined behavior
- Sum types
- Zero dependency binaries
- Can translate C to V automatically
- Fast compile times (as high as 500k LOC/s)
- GC
  - But is optional
- Memory guarantees?

- https://xeiaso.net/blog/v-vaporware-2019-06-23 (from 2019)
- Compile time was exaggerated
- Compiler is a dynamically linked binary
- Compiler and generated code leak memory

- https://xeiaso.net/blog/v-vaporware-2019-06-23 (from 2019)
- Compile time was exaggerated
- Compiler is a dynamically linked binary
- Compiler and generated code leak memory
- https://mawfig.github.io/2022/06/18/v-lang-in-2022.html#summary (from 2022)
- Can create null pointers
- Backend is C, and can generate C w/ undefined behavior
- Array bounds checking is not robust
- Immutability is easily bypassed
- “Pure” functions are meaningless
- Doesn’t prevent global variables
- Performance claims don’t hold up
- V compiler is much slower than advertised
- V’s “autofree” appears to be vaporware

- Simple
- Mutable Value Semantics
  - Stronger theoretical standing
- Fast

- Simple
- Mutable Value Semantics
  - Stronger theoretical standing
- Fast
- Incomplete as of now
  - Plans for generics, stdlib, etc.
- Two papers describing MVS:
  - Implementation Strategies for Mutable Value Semantics
  - Native Implementation of Mutable Value Semantics
  - Used to be called “Val”

- Created by Evan Ovadia
  - “there's at least eleven [memory management] methods”
- Fast
- Memory safety through generational references
- Memory safety through region borrow checking
- Memory safety through single ownership

- Created by Evan Ovadia
  - “there's at least eleven [memory management] methods”
- Fast
- Memory safety through generational references
- Memory safety through region borrow checking
- Memory safety through single ownership
- Not as simple
- Memory safety is *not* never will dereference a bad reference - it’s there’s no undefined behavior when dereferencing a bad reference

- https://verdagon.dev/blog/generational-references
- Using generational references, dereferencing a dead reference is defined to cause a program crash
- Stack objects are singly owned by containing stack frame - can remove GR checks for stack objects
- Heap objects have a single owning pointer - if function owns a heap object, can remove GR check
- Inside “pure” functions, GR checks can be eliminated if passed objects are “prechecked”, thanks to region borrow checking

ATS

- An ML
- Same performance characteristics as C
- Same memory control as C, and…
- Linear + refinement type system verifies safety at compile time

ATS

- An ML
- Same performance characteristics as C
- Same memory control as C, and…
- Linear + refinement type system verifies safety at compile time
- Bats*** crazy syntax / language in general
- Research software
- Steep learning curve

ATS

"A (Not So Gentle) Introduction To Systems Programming In ATS" by Aditya Siram - mpv

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1 , a @ l2 | i : ptr l1, j : ptr l2, s: sizeof_t a):
    (a @ l1, a @ l2 | void) = "ext#swap"
```

Aditya Siram | A (Not So Gentle) Introduction To Syste | September 29, 2017 | 42 / 154

strangeloop

Sept 28-30, 2017

thestrangeloop.com

Should you use any of these?

Should you use any of these?

Should you use any of these?

https://matklad.github.io/2023/03/26/zig-and-rust.html

Interesting Ideas

Vaporware?

There are many more…

- D
- Nim
- Austral
- Jakt
- Hare
- Myrddin
- Lobster
- Compis
- Cone

There are many more…

- D
- **Nim**
- Austral
- Jakt
- Hare
- Myrddin
- Lobster
- Compis
- Cone