

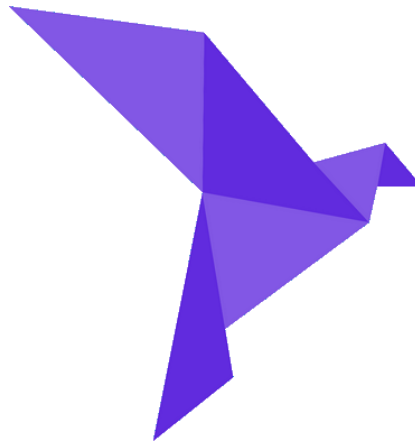
10X faster

taking charge of the compiler
backend

Folkert de Vries

Roc

roc-lang.org



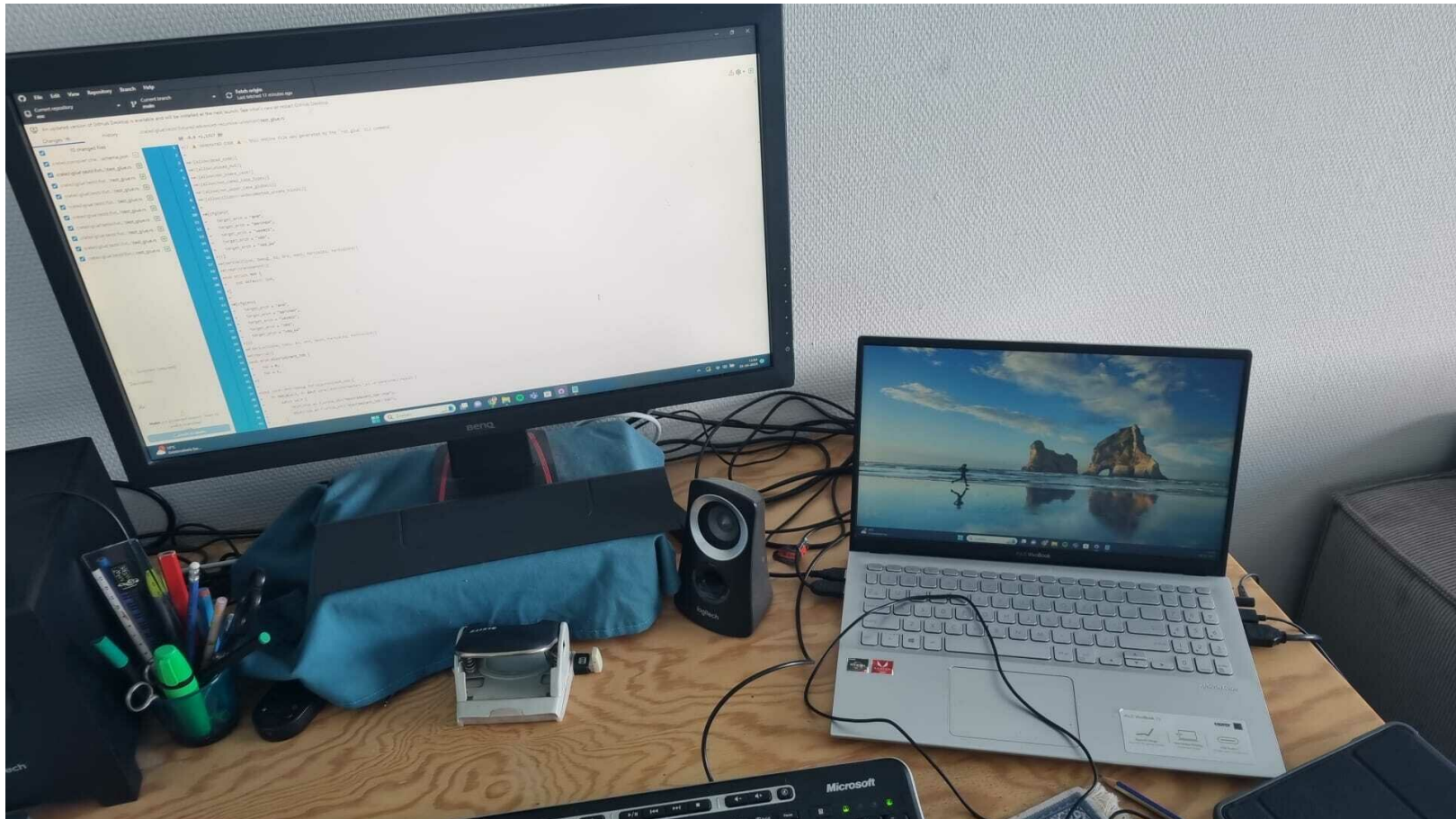
a fast, friendly, functional language

Compiler Feedback is Slow

an anecdote from the raspberry pi



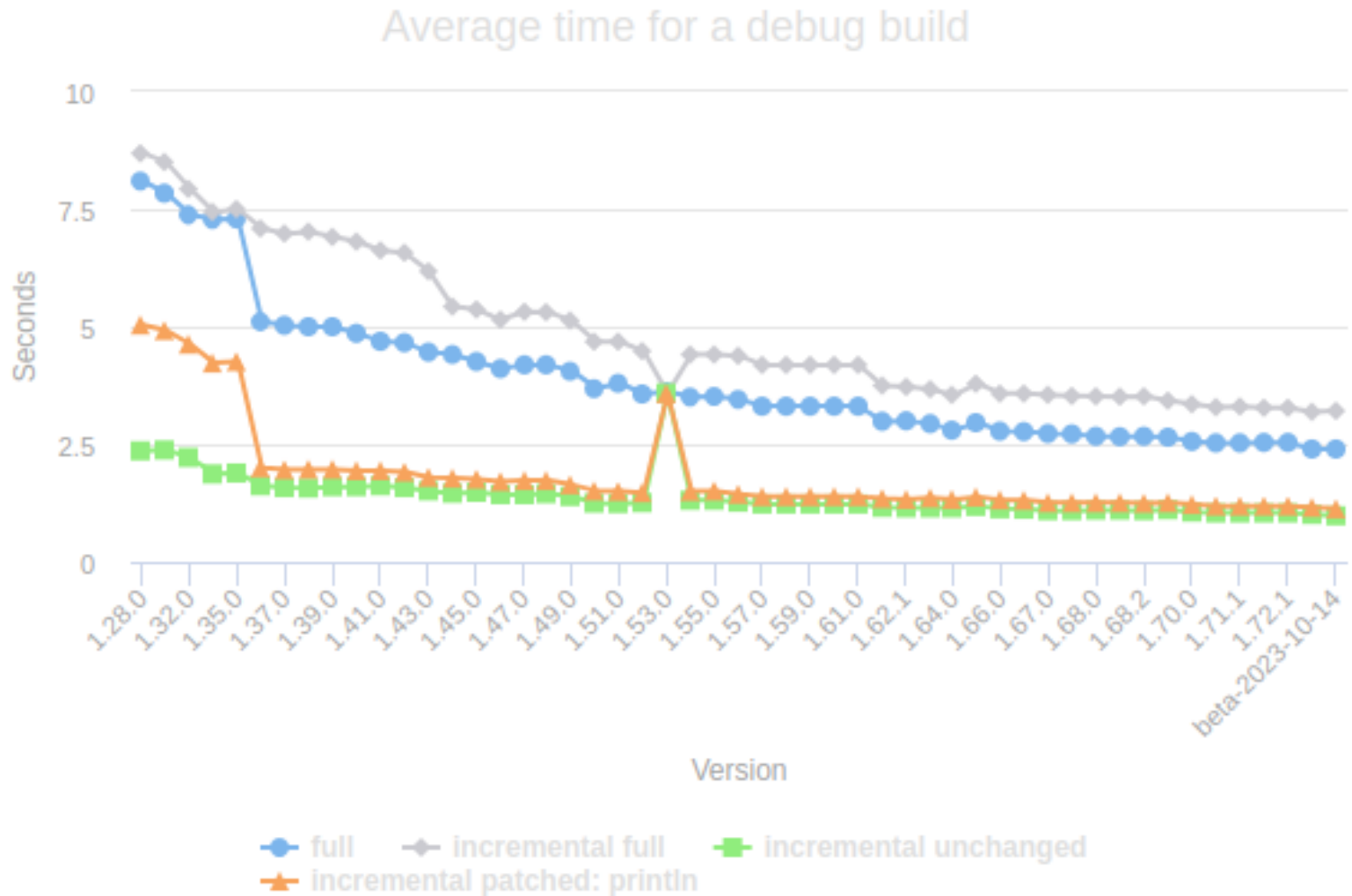
Compiler Feedback is Slow



2X sounds like a lot

but does not change my experience

"rust got faster, though"



"oh sweet summer child"

we don't accept that argumentation elsewhere

error messages

package manager

memory safety

The competition is Python

why is python popular?

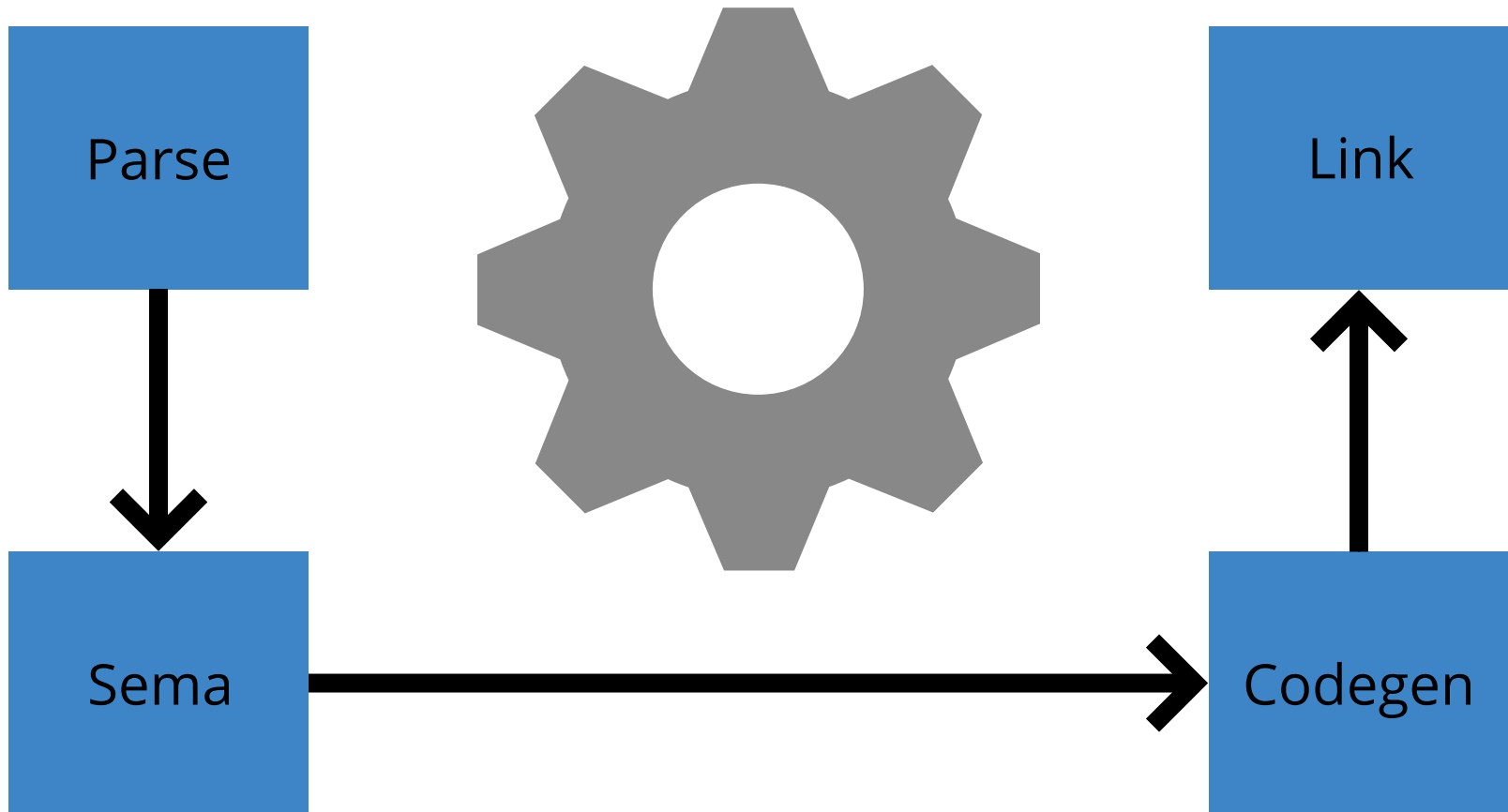
```
>>> a = float("nan")
>>> a == a
False
>>> [a] == [a]
True
>>> [float("nan")] == [float("nan")]
False
>>> []
```


How can we do better?

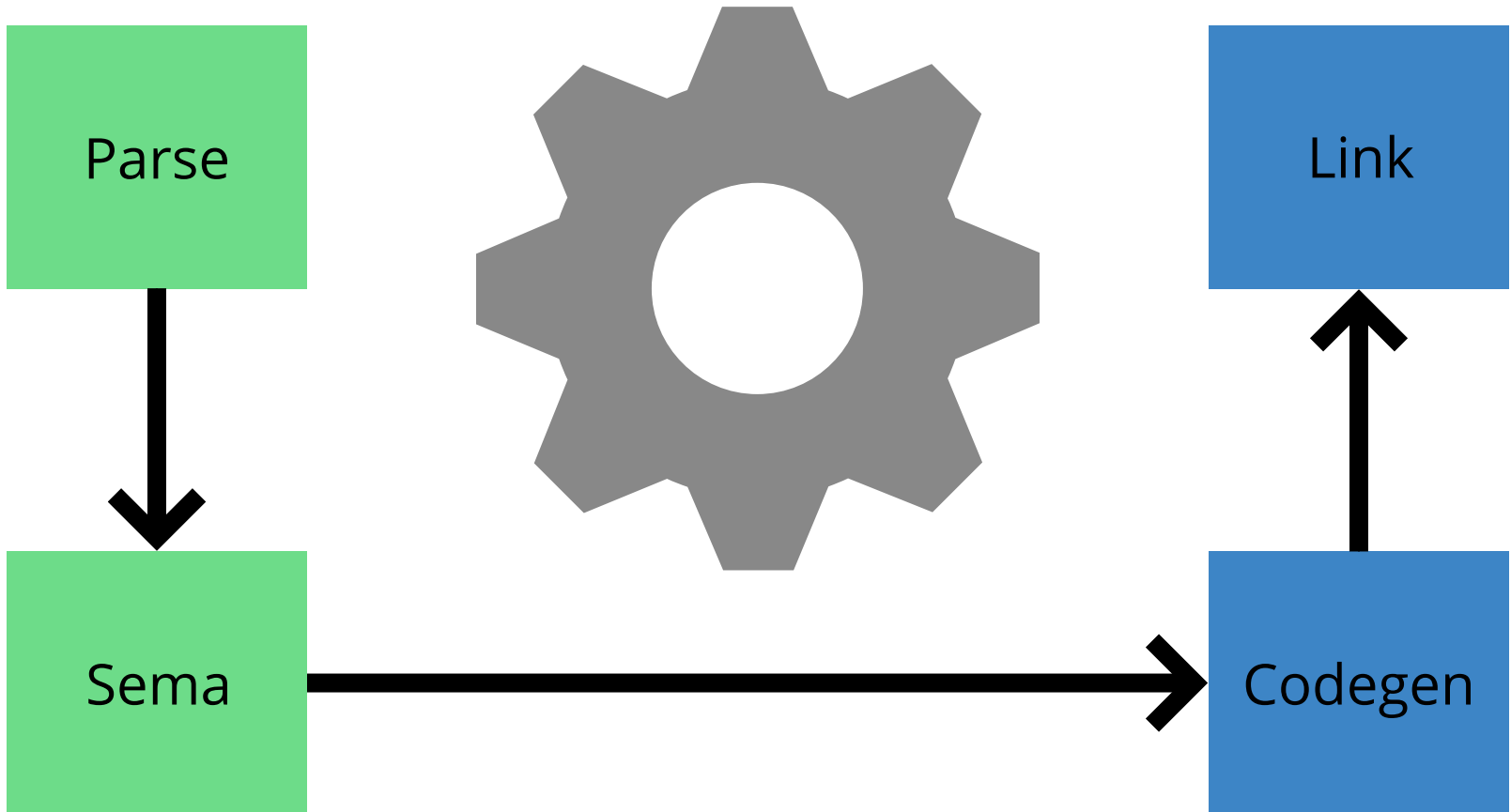
"make it work, then make it fast"
does not work

architecture determines performance

Compilers, how do they work



Engineering Incentives



LLVM



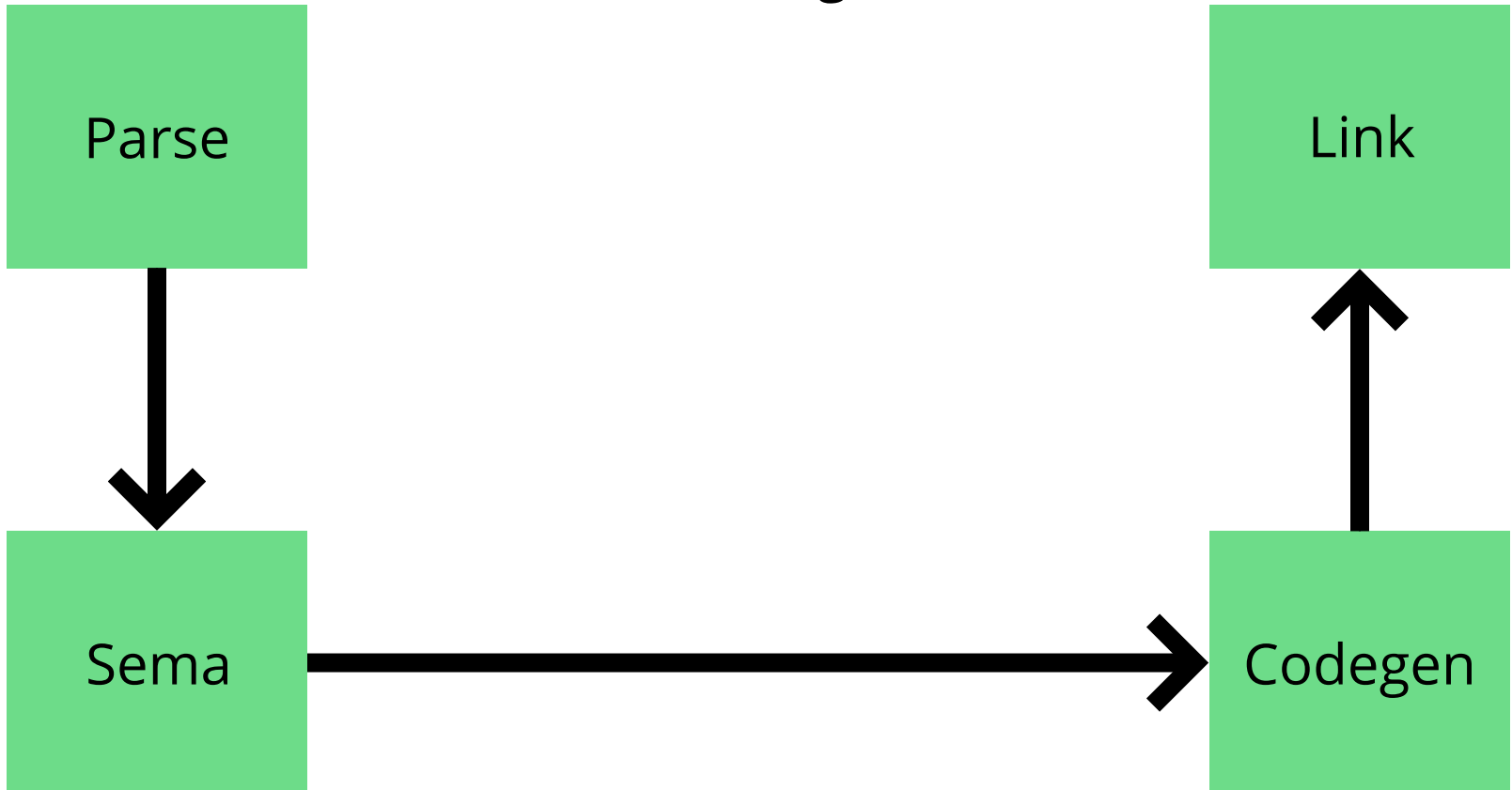
state of the art code generation,
on all the platforms

LLVM

terrible at debug builds

Our Plan

vertical integration



Our ... Challenges

ARM Manual.pdf — Arm Architecture Reference Manual for A-profile architecture

File Edit View Go Bookmarks Help

← → 1 of 11952

Index

Arm Architecture R...	1
Contents	5
▶ Preface	17
Part A: Arm Archite...	33
▶ A1: Introduction to...	35
▶ A2: Armv8-A Archite...	71
▶ A3: Armv9 Architec...	127
Part B: The AArch6...	133
▶ B1: The AArch64 A...	135
▶ B2: The AArch64 A...	151
Part C: The AArch6...	223
▶ C1: The A64 Instru...	225
▶ C2: About the A64 ...	239
▶ C3: A64 Instructio...	247
▶ C4: A64 Instructio...	387

Arm® Architecture Reference Manual
for A-profile architecture

The competition is Python

roc-lang.org/repl

The rockin' Roc REPL

Enter an expression to evaluate, or a definition (like `x = 1`) to use later.

- `Shift-Enter` or `Ctrl-Enter` makes a newline
- `:help`

» `Str.concat "Hello " "World!"`

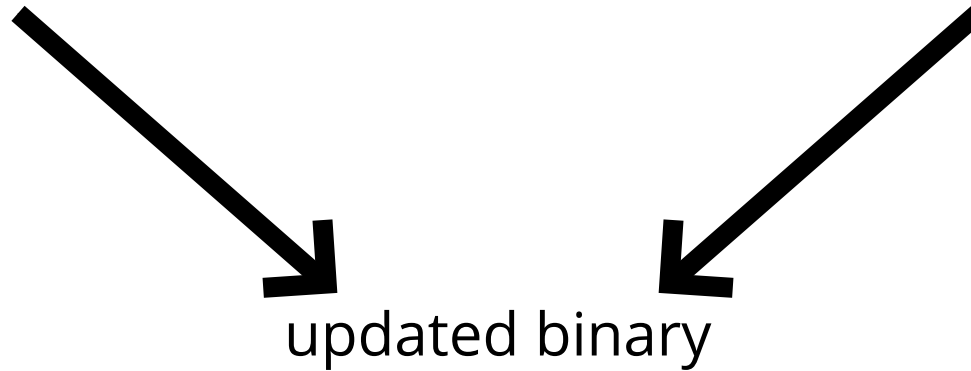
`"Hello World!" : Str`

» Type some Roc code and press Enter.

Surgical Linking

recompiled

precompiled



The competition is Python

We can compile + run hello world
faster than python can print it

Reflections on Rust

Rust deserves fast (re)compiles

Takeaways

be ambitious
(read 12000 page PDFs)

Takeaways

design for performance

orders of magnitude better

Takeaways

build cool things with friends

Expect more from your Tools

Thanks
