

Roto

A fast and safe scripting language for Rust

Oct 09 2025, EuroRust

Terts Diepraam (he/him), NLnet Labs

Terts Diepraam

Software Engineer at NLnet Labs

Organizer of RustWeek

NLnet Labs

Non-profit organization

Been around for 25 years

DNS and routing

E.g. NSD, Unbound, Routinator, etc.

NLnet Labs

Historically all in C

All *new* projects are in Rust

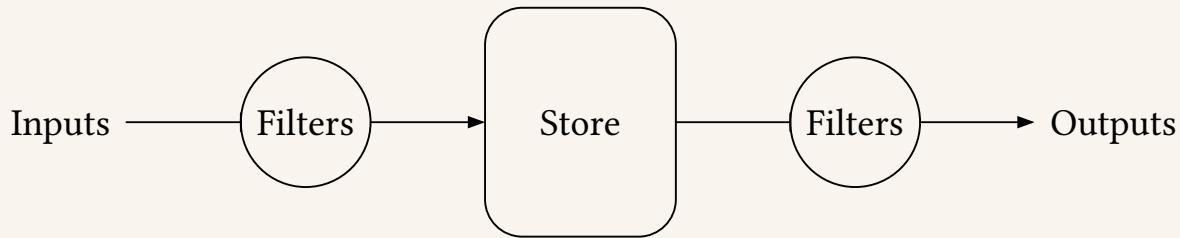
E.g. Cascade, Krill, Routinator, Rotonda and more

Rotonda

A BGP collector written in Rust

read: specialized database

Rotonda: simplified



Solution: scripting language?

Too constrained

Too slow

Dynamically typed

So, being the completely sane developers that we are...

...we made our own

Enter: **Roto**

Roto in a nutshell

Embedded in Rust applications

Statically typed

Friendly error messages

JIT compiled to machine code

How? Cranelift!

Roto compiles to Cranelift IR

Cranelift does the rest!

The unsafest unsafe makes it fast

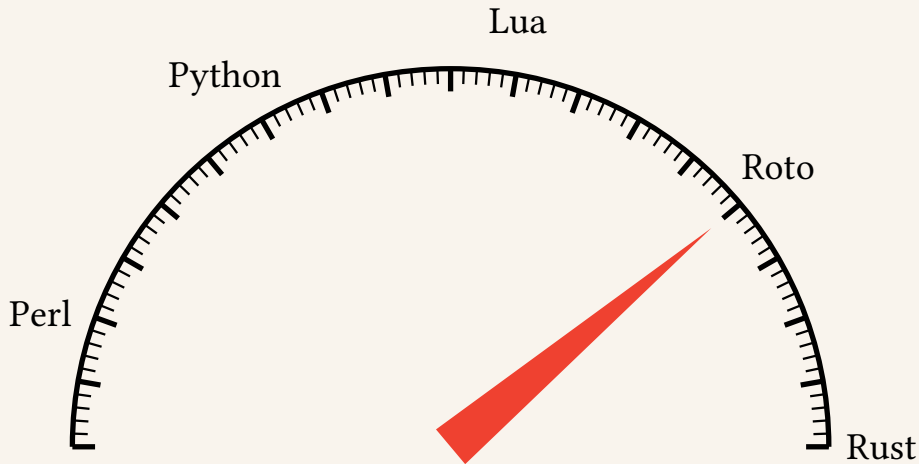
Cranelift gives us just a pointer and a buffer of code.

Transmute `*const u8` to a function pointer

Super-duper unsafe!

Mitigation: Valgrind

Handwavy speed expectations




Example: A simple script

```
fn clamp(x: i32) -> i32 {  
    print(f"Got the value: {x}");  
    if x > 100 {  
        print("It's too big!");  
        x = 100;  
    }  
    x  
}
```

Example: compiling a script

```
let rt = roto::Runtime::new();  
let mut pkg = rt.compile("script.roto");  
let f = pkg.get_function:::<fn(i32) -> i32>("clamp");  
let y = f.call(10);
```


Example: Error messages



```
fn clamp(x: i32) -> i32 {  
    print(f"Got the value: {x}");  
    if x > 100 {  
        print("It's too big!");  
        x = 100.0;  
    }  
    x  
}
```

Example: Error messages

Error: Type error: mismatched types

[script.roto:5:13]

5

x = 100.0;

expected `i32`, found `{float}`

Example: Error messages 2

```
fn clamp(x: i32) -> i32 {  
    print(f"Got the value: {x}");  
    if x > 100 {  
        print("It's too big!");  
        x = 100;  
    }  
    x  
}
```



```
fn clamp() {}
```

Example: Error messages

Error: Type error: item `clamp` is declared multiple times
[script.roto:10:4]

1 fn clamp(x: i32) -> i32 {

 `clamp` previously declared here

10 fn clamp() {}

 `clamp` redefined here

Example: Registration

```
use glam::Vec3; // just some random type
use roto::{Runtime, Val, library};

let lib = library! {
    #[copy] type Vec3 = Val<Vec3>;

    impl Val<Vec3> {
        fn x(self) -> f32 {
            self.x
        }
    }
};

let rt = Runtime::from_lib(lib)?;
```

Example: Registration

```
fn add_x_components(a: Vec3, b: Vec3) -> f32 {  
    a.x() + b.x()  
}
```

Example: Registration

```
let mut pkg = rt.compile("script.roto");  
let f = pkg.get_function("add_x_components");  
  
let a = Vec3::new(3.0, 0.0, 0.0);  
let b = Vec3::new(5.0, 0.0, 0.0);  
let out = f.call(Val(a), Val(b));  
// out == 8.0 (roughly)
```

Registration restrictions

`'static` & `Clone`

Otherwise: `Rc` or `Arc`

No serialization necessary!

[INSERT DEMO HERE]

Documentation website & generator

ROTO

latest ▾

OVERVIEW

Introduction

Goals

Installation

Hello, world!

EMBEDDING ROTO

Setup

Call a Roto function

Expand the runtime

Generate a CLI

REFERENCE

Language Reference

Standard Library

Modules

Types

🏠 / Standard Library / f64

[View page source](#)

f64

type **f64**

The 64-bit floating point type

function **abs**(**x**: f64) → f64

Computes the absolute value of self.

function **ceil**(**x**: f64) → f64

Returns the smallest integer greater than or equal to self.

function **floor**(**x**: f64) → f64

Returns the smallest integer greater than or equal to self.

function **is_finite**(**x**: f64) → bool

Returns true if this number is neither infinite nor NaN.

function **is_infinite**(**x**: f64) → bool

Returns true if this value is positive infinity or negative infinity, and false otherwise.

Current limitations

- No lists
- No `for` loops (only `while`)
- No generics
- And some more

We take this thing seriously!

Backed by a non-profit organization

Integral part of a major product

Free and open source forever

Join us for RustWeek 2026!



May 18-23, 2026 – Utrecht, The Netherlands

See rustweek.org

Links

More about Roto

- github.com/NLnetLabs/roto
- roto.docs.nlnetlabs.nl

Find me online

- terts.dev
- terts@nlnetlabs.nl
- [@mastodon.online@tertsdiepraam](https://mastodon.online/@tertsdiepraam)

Feel free to come up and talk to me!

Slides made with Typst.

Slides, recording & links:



<https://terts.dev/talks/roto-eurorust25>