

Forum:[Hack Forums](#)

Thread: [Languages Written In thei Own Language](#)

Omniscient - December 23, 2025, 11:40 am

I've always wondered how compiled languages are written in their own language. Asked AI about it today. Asked because Rust is written in Rust and I'm like "how the fugg".

Here it is...

AI Quote: Great question. This gets to the heart of how programming languages actually come into existence*.

Short version: they “bootstrap” themselves.

The core idea: bootstrapping

A programming language doesn't start by magically compiling itself.

It starts like this:

1. Version 0 of the compiler is written in some *other language
2. That compiler can compile simple programs in the new language
3. One of those programs is a new compiler written in the new language
4. From then on, the language can compile itself

That's it. No paradox.

Step-by-step example (simplified)

Let's imagine creating a language called `Foo`.

Step 1: Write the first compiler in another language

You write:

``foo-compiler-v0`` in C (or OCaml, etc.)

This compiler:

Is very basic Maybe slow
Supports only part of Foo

But it works.

Step 2: Write a better compiler in Foo

Now you write:

``foo-compiler-v1`` in Foo

You use:

``foo-compiler-v0`` to compile it

Now you have:

A compiler written in Foo

Running as a native executable

Step 3: Replace the old compiler

Once you trust it:

Use ``foo-compiler-v1`` to compile itself

Throw away the original C compiler (or keep it for bootstrapping new machines)

Now Foo is self-hosting.

Why this works logically

There is no infinite loop because:

The **first compiler is external* Self-hosting happens after the language already exists

At runtime, compilers are just programs, not magical objects

It's like:

> You learn English from a teacher,

> then later you teach English in English*.

Real-world examples

Rust

Early compiler written in OCaml Gradually rewritten in Rust

Today: ``rustc`` compiles ``rustc``

C

Early C compiler written in assembly

Later rewritten in C Compiles itself on every platform

Go

Originally written in C Now written in Go

Python

CPython is written in C But tools like PyPy are written in Python

The “trusting trust” problem (important nuance)

Ken Thompson famously pointed out:

If the original compiler is malicious It can insert hidden backdoors into all future compilers

Even if the source looks clean

This is why:

Reproducible builds

Multiple independent compilers Formal verification

matter in serious language ecosystems.

Bottom line

> A language is not born self-aware.

It starts:

as a compiler written in another language becomes powerful enough to reimplement itself then maintains itself forever after

That's how Rust, C, Go, Swift, and most serious languages work.

If you want, I can also explain:

how bootstrapping works across different CPU architectures

how you build the first compiler on a brand-new machine

* or how interpreters differ from compilers in this process

I think that's pretty cool. Just make an initial compiler, create new lang, compile a new compiler in the new lang, and that's it. Simple but works.

m0hx - December 23, 2025, 11:44 am

my brain is not braining :(

the real question is how the first language was created. ?

Omniscient - December 23, 2025, 11:54 am

(Dec 23, 2025 12:44 PM)m0hx Wrote: [\[CITE\]](#) my brain is not braining :(

the real question is how the first language was created. ?

Machine Language, literally zeroes and ones binary.

Code B8 01 00 00 00 ; mov eax, 1

BB 02 00 00 00 ; mov ebx, 2

01 D8 ; add eax, ebx

83 C0 03 ; add eax, 3

89 C1 ; mov ecx, eax

83 E9 01 ; sub ecx, 1

31 D2 ; xor edx, edx

F7 F1 ; div ecx

89 D0 ; mov eax, edx

C3 ; ret

Terry A. Davis - December 23, 2025, 11:38 pm

(Dec 23, 2025 12:54 PM)Omniscient Wrote: [\[CITE\]](#) Machine Language, literally zeroes and ones binary.

```
Code B8 01 00 00 00 ; mov eax, 1
BB 02 00 00 00 ; mov ebx, 2
01 D8 ; add eax, ebx
83 C0 03 ; add eax, 3
89 C1 ; mov ecx, eax
83 E9 01 ; sub ecx, 1
31 D2 ; xor edx, edx
F7 F1 ; div ecx
89 D0 ; mov eax, edx
C3 ; ret
```

You should try redstone one time, it'll help. Seriously, the binary nature shows you how we get to these crazy opcodes from power or no power, it's bonkers sure, but the more you break it down the more it makes sense.

Even Arduino with power stuff, it's not like easy, but it does help understand the crazy abstraction when you start to see logic gates form, then combine them to go to opcodes and more complex gates etc etc etc etc

Omniscient - December 24, 2025, 10:04 am

(Dec 24, 2025 12:38 AM)Terry A. Davis Wrote: [\[CITE\]](#) You should try redstone one time, it'll help. Seriously, the binary nature shows you how we get to these crazy opcodes from power or no power, it's bonkers sure, but the more you break it down the more it makes sense.

Even Arduino with power stuff, it's not like easy, but it does help understand the crazy abstraction when you start to see logic gates form, then combine them to go to opcodes and more complex gates etc etc etc etc

I don't have need for this right now. I don't dabble in it. Just saying some basics for others who may have no understanding of it.

Terry A. Davis - December 25, 2025, 2:22 pm

(Dec 24, 2025 11:04 AM)Omniscient Wrote: [\[CITE\]](#) I don't have need for this right now. I don't dabble in it. Just saying some basics for others who may have no understanding of it.

I feel you homie. Others reading can still take my message and apply to themselves if they choose to.

It's a fascinating world, quite a lot of languages bootstrap themselves, like c#, and a lot of modern features are just wrapping back to primitive c#.

It's like OS Dev, you kinda start with no nothing and just continue stacking the Jenga blocks until you have a suite of things, like UI or networking stuff so you're not needing to continue to remake things.

Good times, good times.

kindproxy.com - January 22, 2026, 7:31 am

This is the best explanation of 'Chicken and Egg' problem in CS I've seen. It's wild to think that Rust used OCaml as its 'mother' before it could stand on its own feet. It's like the language had to undergo a full brain transplant at some point. Thanks for sharing this!