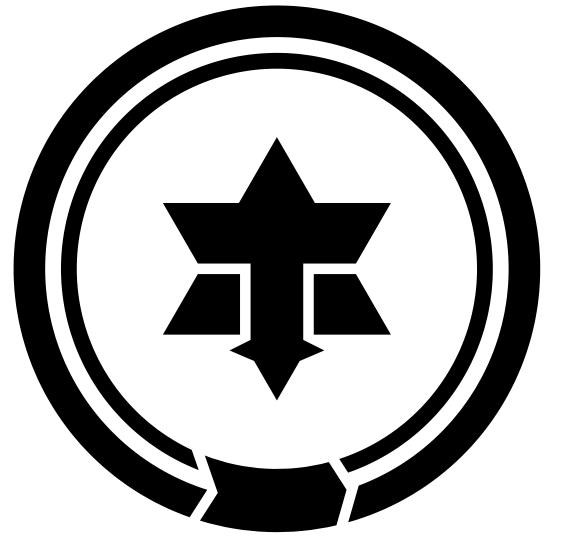


Fitting Rust YJIT into CRuby

Alan Wu

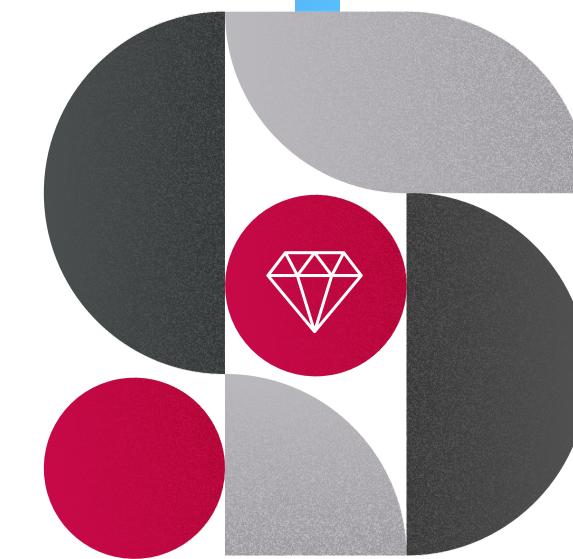


RubyKaigi
令和5年5月
松本市

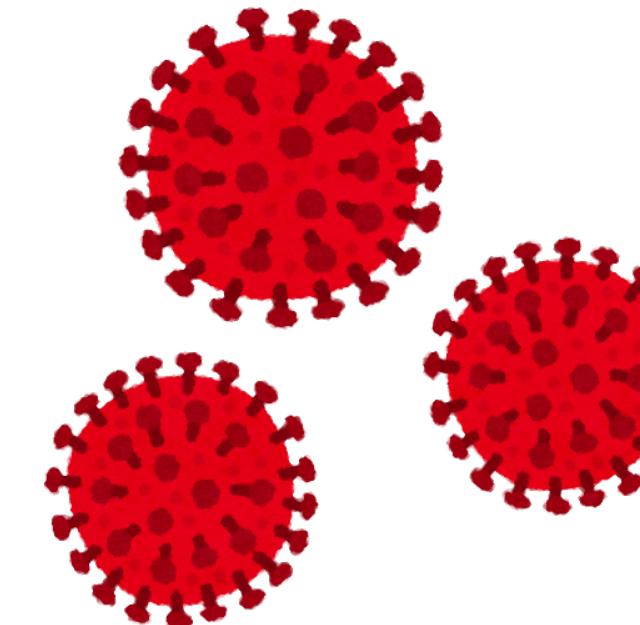
Fitting Rust YJIT into CRuby

- Background YJIT? ワイ JIT? Why JIT?
- Integration challenges the new environment
- Speaking C language learning
- Linking integration



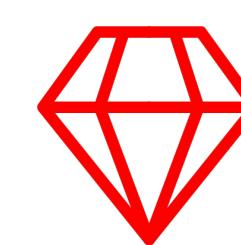


2020

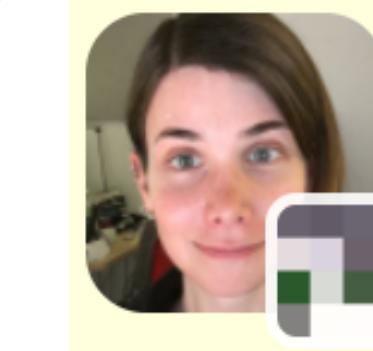


μ JIT

2021



YJIT



Proposal to merge YJIT

Added by [maximecb](#) (Maxime)

2022



Porting YJIT to Rust

Added by [maximecb](#) (Maxime Chevalier-Boisvert)

Faster and leaner

Larger team

AArch64 support



Why Rust?

- We had a lot of learning to do regardless of language choice
- Rust had/has attractive learning resources
- Borrow checker could be used to detect issues in generated code
- Contemporary conveniences
- Awesome tooling out of the box



Typical Ruby users
don't care about Rust

Façade of magic



- Flip a switch, and the program runs quicker!
- There's stuff happening inside, but I'm too busy to look
- I don't want to make *any* changes to my code/workflow
- A build of Ruby with YJIT as the side, please



Chapter 2

Integration Challenges

What is it like in CRuby Land



Don't break the build...

No official binaries on ruby-lang.org

The screenshot shows the official Ruby website at ruby-lang.org. The header features the Ruby logo and the tagline "A PROGRAMMER'S BEST FRIEND". A navigation bar includes links for Downloads, Documentation, Libraries, Community, News, Security, and About Ruby. A search bar is also present. The main content area has a dark background. A red box highlights the "Download Ruby" section, which contains text about getting the latest Ruby distributions and links to "Ruby's License". Below this, a "Ways of Installing Ruby" section lists methods for Linux/UNIX, macOS, and Windows, along with a link to the Installation page. Another red box highlights the "Compiling Ruby — Source Code" section, which discusses installing from source code. To the right, a sidebar titled "Get Started, it's easy!" provides links to Try Ruby!, Ruby in Twenty Minutes, Ruby from Other Languages, and documentation sections like Academic Research, Libraries, and Success Stories. The sidebar also encourages participation in the community through Mailing Lists, User Groups, and Blogs.

Download Ruby

Here you can get the latest Ruby distributions in your favorite flavor. The current stable version is 3.2.2. Please be sure to read [Ruby's License](#).

Ways of Installing Ruby

We have several tools on each major platform to install Ruby:

- On Linux/UNIX, you can use the package management system of your distribution or [third-party tools](#) ([rbenv](#) and [RVM](#)).
- On macOS machines, you can use third-party tools ([rbenv](#) and [RVM](#)).
- On Windows machines, you can use [RubyInstaller](#).

See the [Installation](#) page for details on using package management systems or third-party tools.

Of course, you can also install Ruby from source on all major platforms.

Compiling Ruby — Source Code

Installing from the source code is a great solution for when you are comfortable enough with your platform and perhaps need specific settings for your application.

Get Started, it's easy!

[Try Ruby! \(in your browser\)](#)
[Ruby in Twenty Minutes](#)
[Ruby from Other Languages](#)

Explore a new world...

[Documentation](#)
[Academic Research](#)
[Libraries](#)
[Success Stories](#)

Participate in a friendly and growing community.

[Mailing Lists](#): Talk about Ruby with programmers from all around the world.
[User Groups](#): Get in contact with Rubists in your area.
[Blogs](#): Read about what's happening right now in the



So many options



On Windows 10, you can also use the [Windows Subsystem for Linux](#) to install one of the supported Linux distributions and use any of the installation methods available on that system.

Here are available installation methods:

- [Package Management Systems](#)
 - [Debian, Ubuntu](#)
 - [CentOS, Fedora, RHEL](#)
 - [Snap](#)
 - [Gentoo](#)
 - [Arch Linux](#)
 - [macOS](#)
 - [FreeBSD](#)
 - [OpenBSD](#)
 - [OpenIndiana](#)
 - [Windows Package Manager](#)
 - [Chocolatey package manager for Windows](#)
 - [Other Distributions](#)
- [Installers](#)
 - [ruby-build](#)
 - [ruby-install](#)
 - [RubyInstaller \(Windows\)](#)
 - [Ruby Stack](#)
- [Managers](#)
 - [asdf-vm](#)
 - [chruby](#)
 - [rbenv](#)
 - [RVM](#)
 - [uru](#)
- [Building from source](#)

Past breakages

- Trying to link with `libcapstone`
- Assuming echo "\n" prints a new line



Past breakages

- Trying to link with libcapstone
- Assuming echo "\n" prints a new line

```
diff --git a/configure.ac b/configure.ac
index 603cbebd03..0357a7eacf 100644
--- a/configure.ac
+++ b/configure.ac
@@ -1268,15 +1268,6 @@ AC_CHECK_LIB(crypt, crypt)      # glibc (GNU/Linux, GNU/Hur
 AC_CHECK_LIB(dl, dlopen)        # Dynamic linking for SunOS/Solaris and SYSV
 AC_CHECK_LIB(socket, shutdown)  # SunOS/Solaris
```

```
-AS_IF([test -n "${PKG_CONFIG}" && ${PKG_CONFIG} --exists capstone], [
-  CAPSTONE_CFLAGS=`${PKG_CONFIG} --cflags capstone`
-  CAPSTONE_LIB_L=`${PKG_CONFIG} --libs-only-L capstone`
-  LDFLAGS="$LDFLAGS $CAPSTONE_LIB_L"
-  CFLAGS="$CFLAGS $CAPSTONE_CFLAGS"
-])
-
-AC_CHECK_LIB(capstone, cs_open) # Capstone disassembler for debugging YJIT
-
```

```
dnl Checks for header files.
AC_HEADER dirent
dnl AC_HEADER_STDC has been checked in AC_USE_SYSTEM_EXTENSIONS
```

```
diff --git a/configure.ac b/configure.ac
index 174e395cd6..ae560297ad 100644
--- a/configure.ac
+++ b/configure.ac
@@ -3755,7 +3755,7 @@ AS_IF([test "$RUSTC" != "no"],
)
dnl Fails in case rustc target doesn't match ruby target.
dnl Can happen on Rosetta, for example.
AS_IF([echo "#[cfg(target_arch = \"$YJIT_TARGET_ARCH\")][-\n-] fn main() { let x = 1; format!(\"{x}\"); }" |
$RUSTC --emit asm=/dev/null 2>/dev/null],
[YJIT_RUSTC_OK=yes]
)
```



Minimalism

Just make Ruby

- For incremental builds
- GNU make, bsdmake, and nmake
- Has its own language. There is a small feature set that works everywhere
- Rust's cargo gives incremental build out of the box
- Initial setup called cargo unconditionally every time





not_gmake.mk

4 months ago



yjit.mk

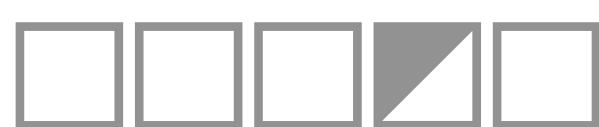
last month



Chapter 3

Speaking C

using the lingua franca



```
$ tokei --compact --exclude ext --exclude spec -t=C,Rust
```



C
~220K lines



Rust
~21K lines

7.18.1.1 Exact-width integer types

- 1 The typedef name **intN_t** designates a signed integer type with width N , no padding bits, and a two's complement representation. Thus, **int8_t** denotes a signed integer type with a width of exactly 8 bits.
- 2 The typedef name **uintN_t** designates an unsigned integer type with width N . Thus, **uint24_t** denotes an unsigned integer type with a width of exactly 24 bits.
- 3 These types are optional. However, if an implementation provides integer types with widths of 8, 16, 32, or 64 bits, no padding bits, and (for the signed types) that have a two's complement representation, it shall define the corresponding typedef names.

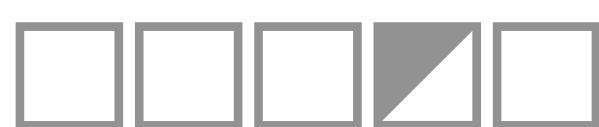


```
rb_define_method(rb_mKernel, "inspect", rb_obj_..., 0);
rb_define_method(rb_mKernel, "singleton_methods", rb_obj_..., -1);
```

```
Kernel.instance_method(:singleton_methods).arity
=> -1
```



```
// Copy arguments from locals
for (int32_t i = 0; i < bf->argc; i++) {
    x86opnd_t stack_opnd = ctx_stack_opnd(ctx, bf->argc - i - 1);
    x86opnd_t c_arg_reg = C_ARG_REGS[2 + i];
    mov(cb, c_arg_reg, stack_opnd);
}
```



```
error[E0277]: the type `[usize]` cannot be indexed by `i32`
```

```
--> rust-signed-index-build-error.rs:7:37
```

```
7 |     let c_arg_reg = C_ARGS_REGS[2 + i];
```

```
    |          ^^^^^ slice indices are of type `usize` or ranges of `usize`
```

```
= help: the trait `SliceIndex<[usize]>` is not implemented for `i32`
```

```
= help: the trait `SliceIndex<[T]>` is implemented for `usize`
```

```
= note: required for `<[usize]>` to implement `Index<i32>`
```

```
error: aborting due to previous error
```

For more information about this error, try `rustc --explain E0277`.



```
// Copy arguments from locals
for (int32_t i = 0; i < bf->argc; i++) {
    x86opnd_t stack_opnd = ctx_stack_opnd(ctx, bf->argc - i - 1);
    x86opnd_t c_arg_reg = C_ARG_REGS[2 + i];
    mov(cb, c_arg_reg, stack_opnd);
}
```

```
fn gen_invokebuiltin(
    jit: &mut JITState,
    asm: &mut Assembler,
    _ocb: &mut OutlinedCb,
) -> Option<CodegenStatus> {
    let bf: *const rb_builtin_function = jit.get_arg(0).as_ptr();
    let bf_argc: usize = unsafe { (*bf).argc }.try_into().expect("non negative argc");
```



References

T ***restrict**

&mut T

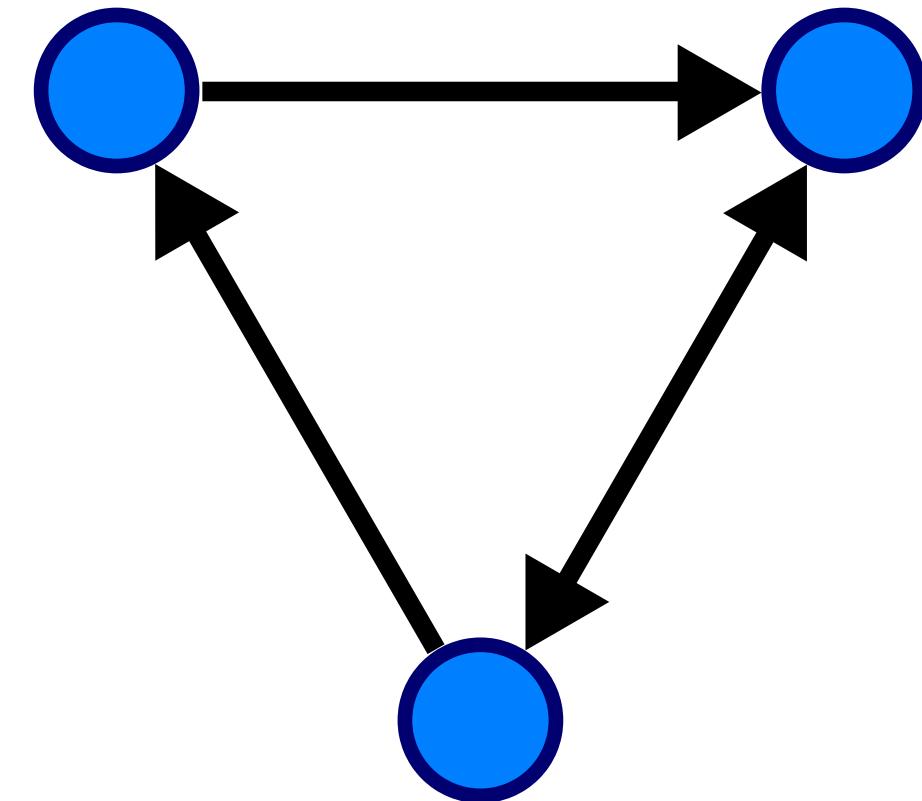
const T *

&T



Dealing with restrictions

- We have graphs in YJIT
- Tried RefCell. It ate memory and added crash risks
- Studied the semantics of `&mut` and `&` through communications from T-opsem (Team Operational Semantics)
- Currently using `unsafe` (hopefully correctly!)



T ***restrict**

const T *

&mut T

&T



When the rules are muddy

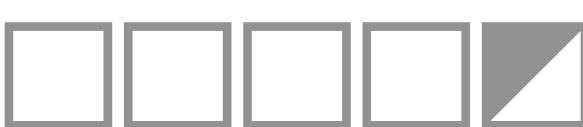
- Find first party (or close to first party) sources
- Read discussions
- Ask questions!
 - It does take effort to formulate good questions
 - But trust that there are people willing to help if you ask nicely 😊



Chapter 4

Linking

connecting machine code



Linking "link-editing"

From the first edition of Unix Programmer's Manual

11/3/71

LD (I)

NAME **ld** -- link editor

SYNOPSIS **ld** [-usaol] name1]

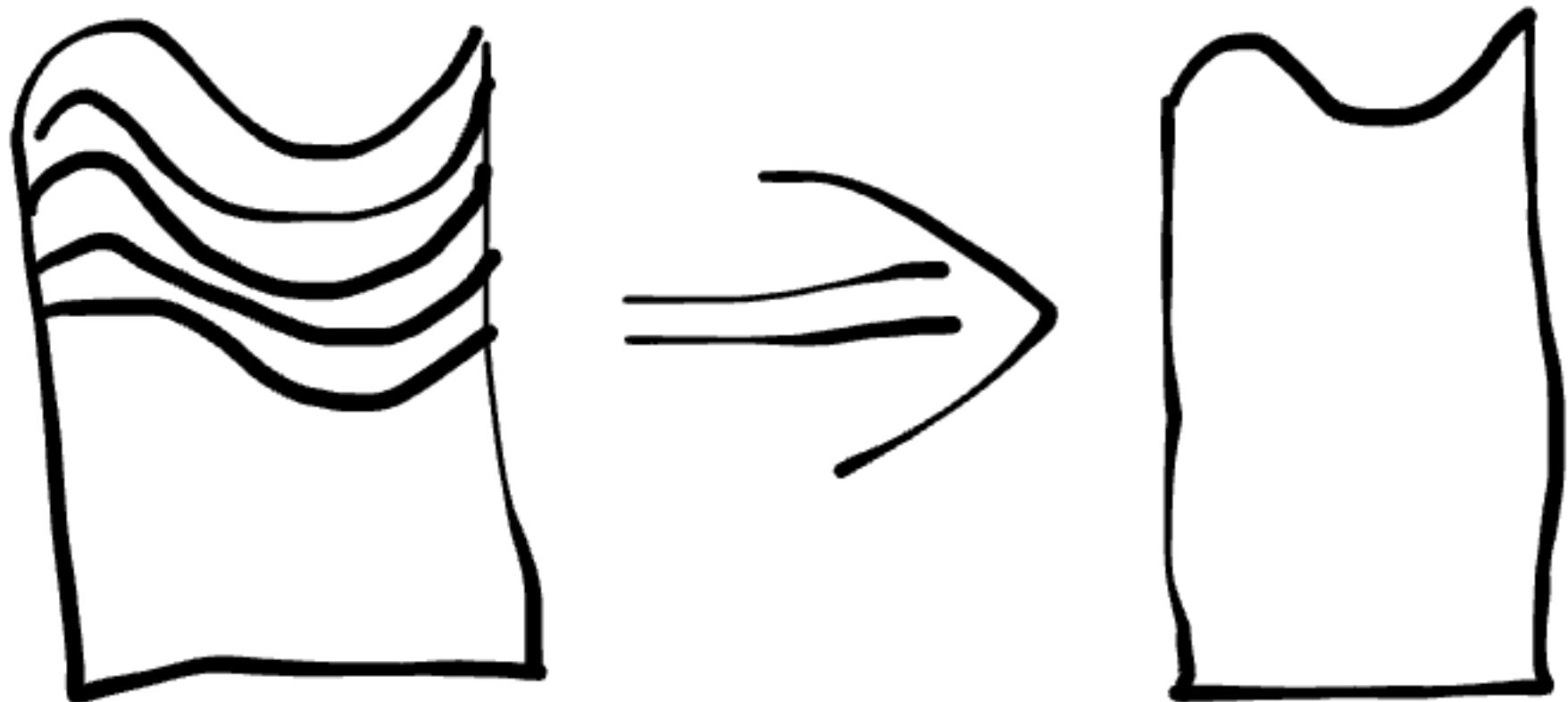
DESCRIPTION **ld** combines several object programs into one; resolves external references; and searches libraries. In the simplest case the names of several object programs are given, and **ld** combines them, producing an object module which can be either executed or become the input for a further **ld** run.

<https://www.bell-labs.com/usr/dmr/www/1stEdman.html>



The link editing operation

- Several files in, a single file out
- Connect code compiled across spacetime
- Why link at all?
 - Enables incremental builds
 - A way to distribute libraries
 - Linkers and object file format

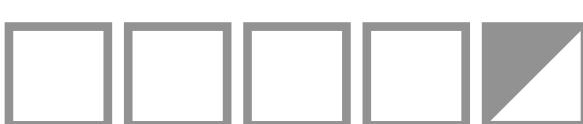


x64 CALL

e8 00 00 00 00

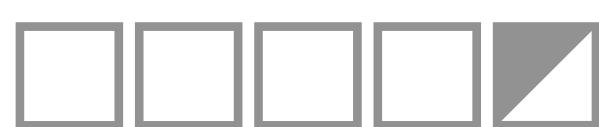
instruction identifier

call target address (relative)



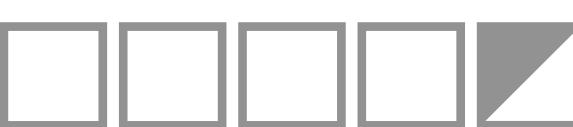
```
struct [REDACTED] {
    content: Content,
    [REDACTED]
}
```

```
enum Content {
    Undefined,
    Defined(Address)
}
```



```
struct [REDACTED] {
    content: Content,
    name: String
}
```

```
enum Content {
    Undefined,
    Defined(Address)
}
```



```
struct [REDACTED] {
    content: Content,
    name: String
}
```

```
struct [REDACTED] {
    where_to_patch: Address,
    what_to_patch_to:
}
```

```
enum Content {
    Undefined,
    Defined(Address)
}
```



```
struct Symbol {  
    content: Content,  
    name: String  
}
```

```
enum Content {  
    Undefined,  
    Defined(Address)  
}
```

```
struct Relocation {  
    where_to_patch: Address,  
    what_to_patch_to: Symbol  
}
```



1839 C. A. HARRIS *Dental Art* xxi. 348 The obtaining of a model of the alveolar ridge, or ridges, when one for each jaw is required, though apparently very easy, is nevertheless often attended with some difficulty. 1857 *Brit. Jnl. Dental Sci.* I. 579/1 Mr. Saunders thought that the plan of bending down the front part of the model could be fatal to a correct impression. 1917 F. A. PEESO *Crown & Bridge-Work* vii. 140 When the plaster for the impression has been tinted, the impression and model are easily distinguished by the difference in coloring. 1938 *Dental Rec.* LVIII. 14, I think, from a study of the original models, that there had probably been pyorrhœa for some years. 1940 [see IMPRESSION sb. 2 e]. 1973 D. H. ROBERTS *Fixed Bridge Prostheses* v. 66 Only one model can be poured from each impression.

e. A simplified or idealized description or conception of a particular system, situation, or process (often in mathematical terms: so **mathematical model**) that is put forward as a basis for calculations, predictions, or further investigation.

Cf. sense 1 c.

1913 N. BOHR in *Phil. Mag.* XXVI. 1 To explain the results of experiments on scattering of α rays by matter Prof. Rutherford has given a theory of the structure of atoms. According to this theory, the atoms consist of a positively charged nucleus surrounded by a system of electrons [etc.]. Great interest is to be attributed to this atom-model.

[i.e. the mould maker] then pours t over the [clay] model until the up thickness. 1856 *Eng. Cycl., Biogr* contents of his studio included models, casts of all his chief work

6. **Plastering.** A tool for having a pattern in profile upon the plaster by working and forwards. Cf. MOULD.

1825 J. NICHOLSON *Operat. Me* tools of the plasterer consist of . . and wood models. *Ibid.*, The running plain mouldings, cornic § 2233.

II. Type of design.

7. Design, structural type form; pattern, build, n structures.

1597 HOOKER *Eccl. Pol.* v. xiv .. then if some King should bu modell of Salomons palace. 16 Trav. 46 This Town is .. bui model. 1698 FRYER *Acc. E. I* that are for this Voyage are hu both the Name and Model o *Gazetteer* (ed. 2), Putney .. b model with that of Fulham.

```
struct Symbol {  
    content: Content,  
    name: String  
}
```

```
enum Content {  
    Undefined,  
    Defined(Address)  
}
```

- >0 Undefined, 0 Defined => "undefined symbol error"
- >1 Defined with the same name => "duplicate symbol error"
- Disassembly for calls seem to go nowhere => use `objdump -r` to see relocations



```
struct Symbol {  
    content: Content,  
    name: String  
}
```

```
enum Content {  
    Undefined,  
    Defined(Address)  
}
```

```
struct Relocation {  
    where_to_patch: Address,  
    what_to_patch_to: Symbol  
}
```

- >0 Undefined, 0 Defined => "undefined symbol error"
- >1 Defined with the same name => "duplicate symbol error"
- Disassembly for calls seem to go nowhere => use `objdump -r` to see relocations



Closing

- Fitting in is a continuous process
- Fitting in is a collaborative process
 - Thanks to nobu, k0kubun, ko1, mame, Maxime, Ralf Jung, and everyone who helped directly or indirectly!
- First party resources such as man pages and the C standard are surprisingly intelligible
- Asking questions takes efforts and can be scary, but people want to help
- Build understanding by building small mental models and refining them

Thanks!

