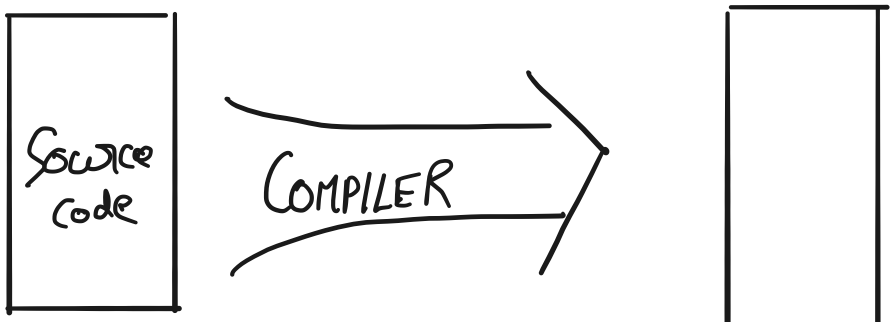
CS 202 – Reflections on Trusting Trust

WHY?

- Cool hack
- TRUST IN SECURITY

COMPILERS



Format A

Often Text

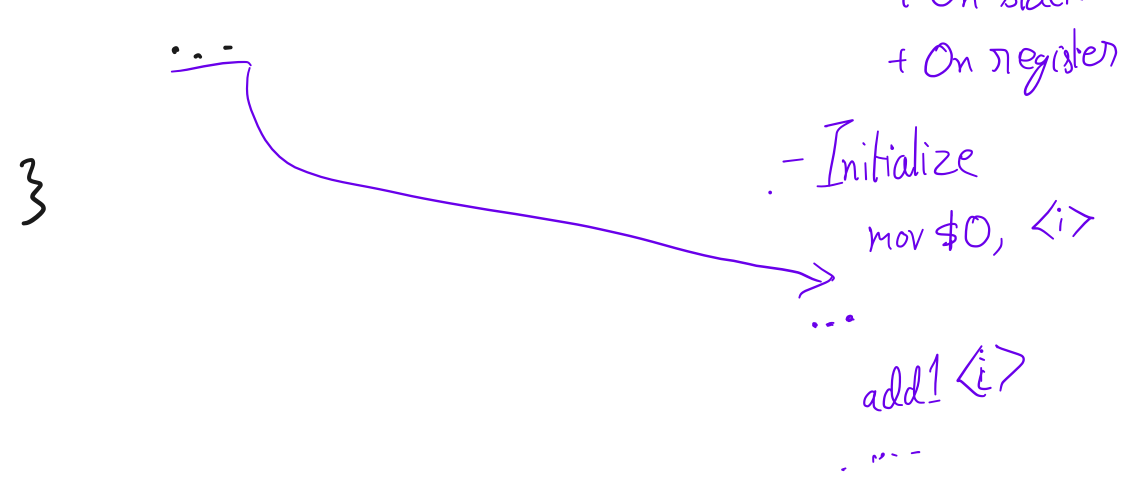
FORMAT B

(elf exe wasm) ...

Just a program

```
for (int i=0; i<2; i++) {
```

- Allocate variable on stack

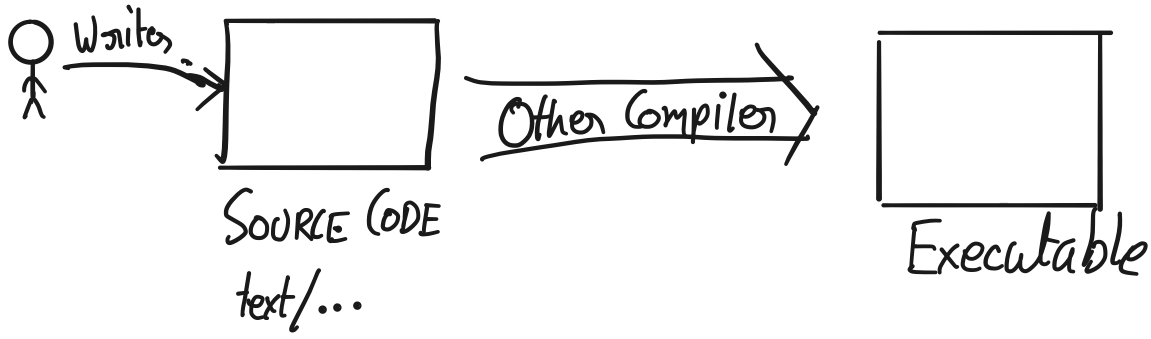


Lots of lookup

- What to generate for constructs
- How to interpret special tokens

\n, \r, \t, ==, →, ...

Building A Compiler

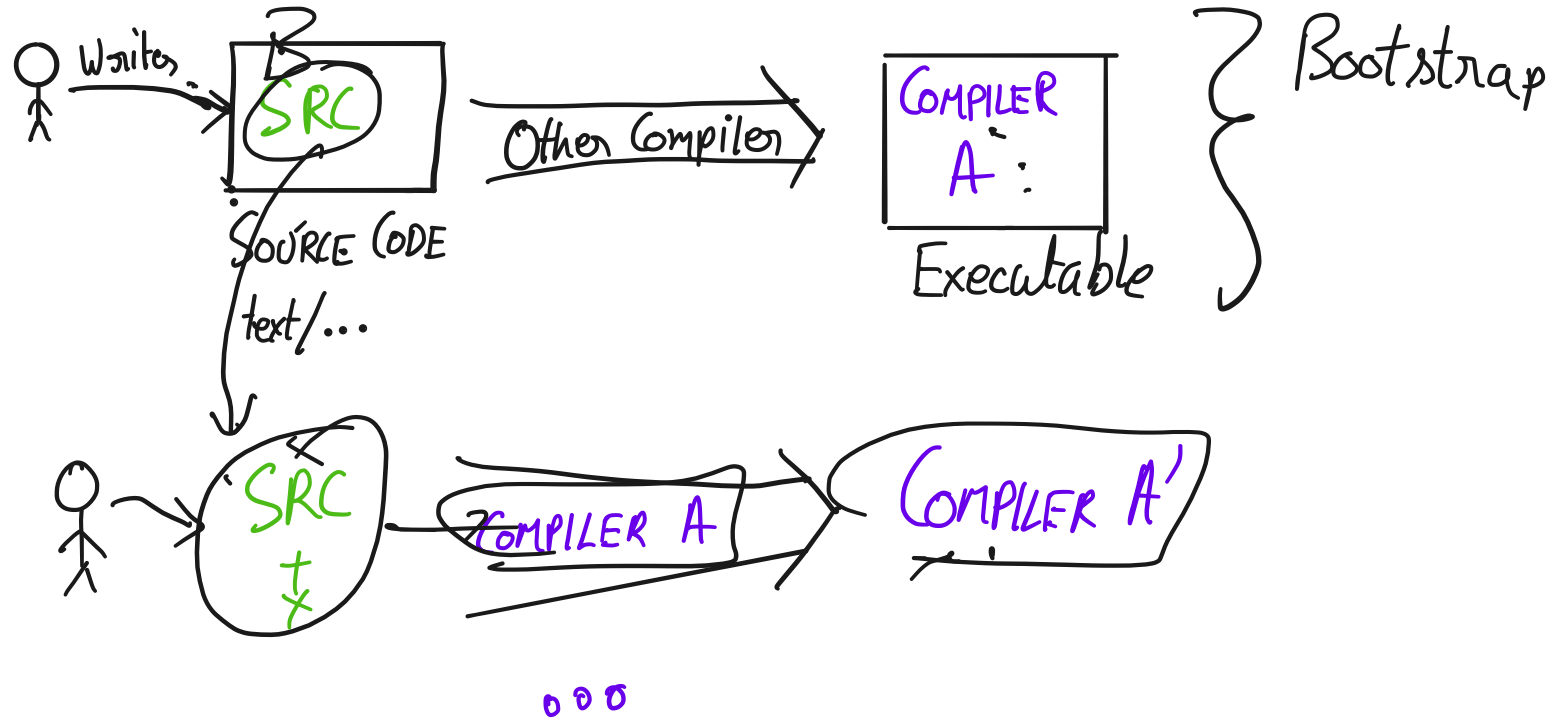


Adding a new feature?

text/...

Executable

SELF-HOSTING COMPILER



Very common

1. gcc
2. clang
3. Rust
4. Go
5. ...

Why? Good test case for compiler capabilities

Simplifying life

- Remember a lot of compiler writing is making a lookup table

(+ a b) ⟹ a = a+b ⟹

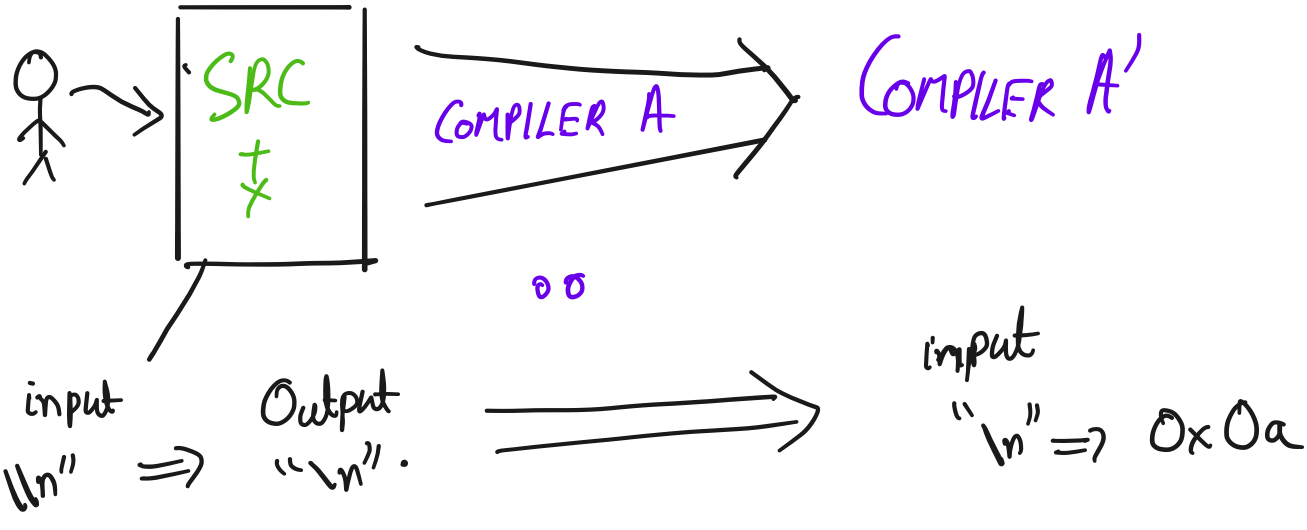
~~a = a+b~~ ⟹ add <a><b>

if (strcmp(token, "\n")
    printf("\n")

\n ⟹ 0x0a

...

- But the compiler used to build the compiler already knows a lot of values



Q: What dictates - compiler behavior (semantics)?

- Code? ✓
- Compiler? ✓

Can you trust code you can read

```
#include <stdio.h>
```

```
#include <stdio.h>
```

```
int main(...) {
    printf("Hello world\n");
}
```

a.c

cc

a.out

```
int main(...) {
    printf("Hello world\n");
    execv("/usr/bin/evil");
}
```

b.c

cc

b.out

```
#include <stdio.h>
int main(...) {
    printf("Hello world\n");
}
```

a.c

cc

a.out

```
#include <stdio.h>o
int main(...) {
    printf("Hello world\n");
}
```

b.c

cc

b.out

↓ Runs /usr/bin/evil

cc.c

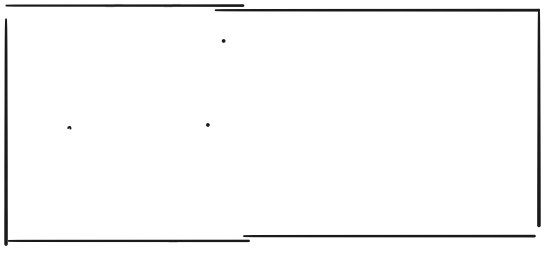
```
if (o in file):
    add(execv("/usr/bin/evil"))
```

cc∅.c

```
if (file is cc.c)
    add (" if (file is cc.c")
    :)
if (file contain ∘)
    add (...)
```

⇒ cc∅

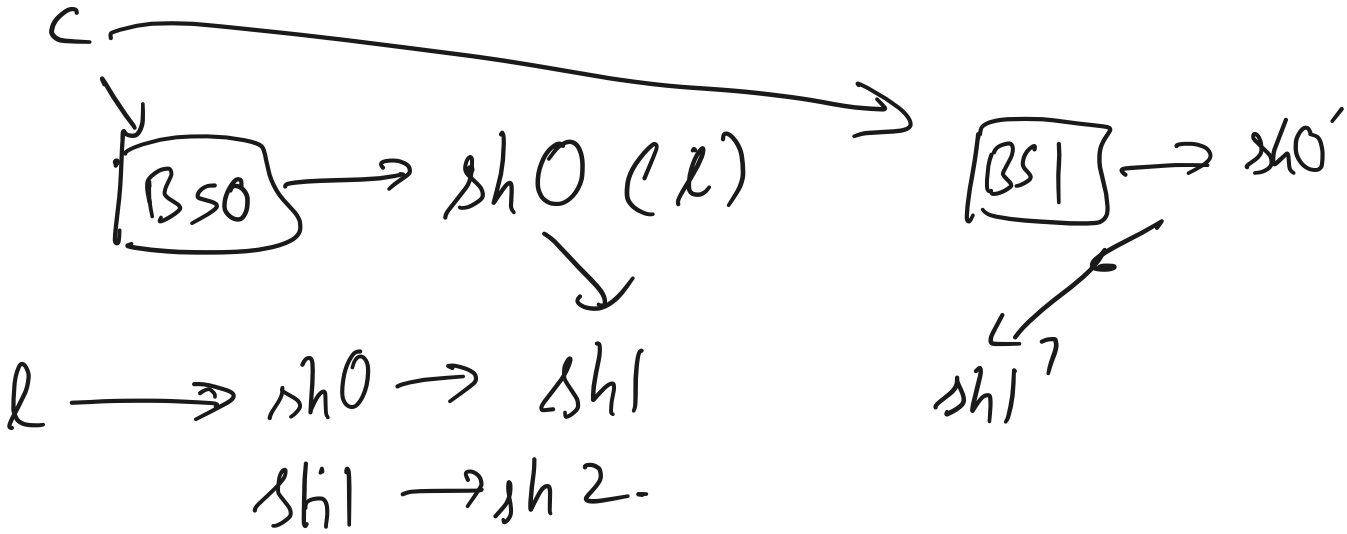
cc.c



⇒ cc

Is this ever really a problem?

Deterministic builds & double compilation



Software Supply Chain