TEACHING OLD COMPILERS NEW TRICKS

Transpiling C++17 to C++11

Tony Wasserka
@fail_cluez

Bellevue
25 September 2018

WHO AM I?

- Freelancer in embedded systems development
- Focus: Low-level & Type-safety
- Side projects: Game console emulators

PPSSPP Dolphin Citra

- Twitter: @fail_cluez
- GitHub: neobrain
- neobrain.github.io

C++14/17: WHY?

- Functionality
- Express programmer intent
- Ergonomic metaprogramming
- Convenient syntax sugar

POTENTIAL

- Readability: "Almost like writing python"
- Lower entry barrier for metaprogramming
- More type-safety ⇒ lower maintenance cost

Check out the C++17 Tony tables

THE PROBLEM

C++17

```
int get_mask() {
  return 0b1000'0000;
}
```

gcc 4.9 $\longrightarrow$

Assembly

```
get_mask:
  mov eax, 128
  ret
```

C++17

```
int get_mask() {
  return 0b1000'0000;
}
```

gcc 4.8 $\longrightarrow$

~~A sembl y~~

```
get_ma
  mov        128
  r
```

SOLUTIONS

- Upgrade to a newer compiler
  - Not always an option
- Stick with older versions of C++
  - Higher cost of maintenance
  - Higher risk of bugs in production
- Use Clang-from-the-Future

CLANG-FROM-THE-FUTURE

- New libclang-based tool
- Preprocessor for a proper compiler
- Automatic conversion of C++17 to C++11

USUAL BUILD PIPELINE

$$\text{Source} \longrightarrow \text{C++ compiler} \longrightarrow \text{Linker} \longrightarrow \text{Executable}$$

CFTF-ENHANCED PIPELINE

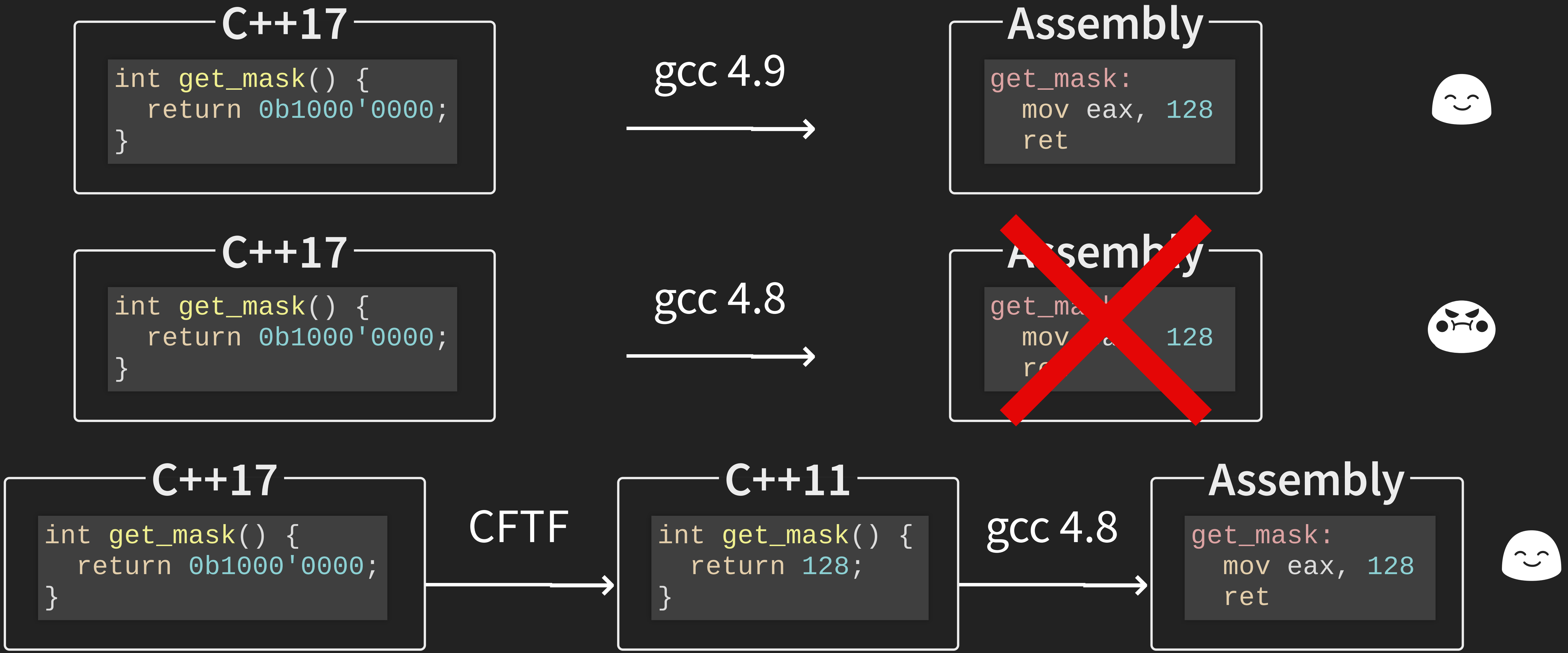
Source ⟶ C++ compiler ⟶ Linker ⟶ Exec.

CFTF-ENHANCED PIPELINE

$$\begin{tikzcd} \text{Source} \arrow[r] & \underset{\text{CFTF}}{\boxed{\text{build AST} \longrightarrow \text{C++11 Source}}} \arrow[r] & \underset{\text{Frontend Compiler}}{\boxed{\text{C++ compiler} \longrightarrow \text{Linker} \longrightarrow \text{Exec.}}} \end{tikzcd}$$

CFTF = Black box precompilation step!

A SOLUTION



C++17

```cpp
int get_mask() {
  return 0b1000'0000;
}
```

gcc 4.9 → Assembly

```asm
get_mask:
  mov eax, 128
  ret
```

C++17

```cpp
int get_mask() {
  return 0b1000'0000;
}
```

gcc 4.8 → ~~Assembly~~

```asm
get_ma
  mov         128
  r
```

C++17

```cpp
int get_mask() {
  return 0b1000'0000;
}
```

CFTF → C++11

```cpp
int get_mask() {
  return 128;
}
```

gcc 4.8 → Assembly

```asm
get_mask:
  mov eax, 128
  ret
```

COMPONENTS

- Command-line interface
- AST visitors (via libclang)
- Rewrite engine
- Template specializer
- Test suite

EXAMPLES

- If-init
- Structured bindings
- Auto return type deduction
- If-constexpr
- Fold expressions

IN PRACTICE

```
cftf -frontend-compiler=/usr/bin/g++ input.cpp
```

Easy integration into your build pipeline:

- Make:

  ```
  CXX=/usr/local/bin/cftf CXX_FLAGS="-frontend-compiler=/usr/bin/g++" make
  ```

- CMake:

  ```
  CXX=/usr/local/bin/cftf cmake -DCMAKE_CXX_FLAGS="-frontend-compiler=/usr/bin/g++
  ```

- Other setups may need some creativity

USE CASES

- Early adoption/evaluation of new standards
- Use of C++17 libraries in C++11 setups
- Ports to legacy platforms
- Seeing what the compiler sees
  For that, also check out C++ Insights

CURRENT STATUS

Published on GitHub

- Usable drop-in for gcc/clang on Linux
  Patches for Windows/macOS welcome!
- Small initial set of supported C++14/17 features
  (correctness first, then features)
- Try it out & report issues!

FUTURE

- More rewriting rules ⇒ What do people want most?
- C++2a input support (contracts, concepts, …)
- C++03 output support (move semantics, initializer lists, …)
- Better test coverage (hana, range-v3, …)
- Full C preprocessor support
- Better debugging experience

Your wishes?

SUMMARY

- Compile C++14/17 on an old compiler
- Early stage, but functional drop-in preprocessor
- Easy integration into existing toolchains
- No source code changes needed

THANKS!

github.com/neobrain/cftf

@fail_cluez

neobrain

(const west const best!)