C++20 Modules & Header Files

Florian Weimer, Red Hat
Nathan Sidwell, Facebook

Modules are coming,
Header-units are a thing
What does that suggest?

Header Units

- Similar to PCH, but more
  - import <stdio.h>; // explicit import
  - #include <stdio.h> // might be turned into import
- Impervious to macros from importer
- Not all header files can be header units

Building a Header Unit

- g++ -x c++-header -fmodules-ts path/to/header
- g++ -x c++-system-header -fmodules-ts stdio.h
- Generates a Compiled Module Interface (CMI)
  - In a gcm.cache directory
  - Locked to compiler build (hopefully major[.minor] in future)
  - A cached artifact – not a distributable thing.
- No object file

Tricky Bit

- Header inclusion allows multiple definitions to exist in a single *program* – that’s how headers work
  - Classes, Inline functions, templates
  - Relies on One Definition Rule (ODR)
- Header units may declare &| define the same entity
  - How do we know if two namespace-scope declarations are for the same thing?
  - class bob; // easy, it has a name!

Unnamed Things Bad

- // header-a
  enum { FALSE, TRUE};
- // header-b
  enum { FALSE, TRUE};
- // header-bad
  enum { FALSE, TRUE, FILE_NOT_FOUND = -1};
- C: those consts have signed or unsigned type
- C++: those consts have their enum’s type
  - Prior to C++20 all those were different types

Transitive knowledge

- Header-units can become known to the compiler via named modules
  - If two header units provide conflicting definitions, bad things will happen …
  - … even if no TU directly imported both units

-
  ```
  // kernel I32LP64
  struct X {
    unsigned long long m;
  };
  ```

-
  ```
  // glibc I32LP64
  struct X {
    unsigned long m;
  }
  ```

-
  ```
  export module foo;
  import <kernel>;
  …
  ```

-
  ```
  export module bar;
  import <glibc>;
  …
  ```

-
  ```
  import foo;
  import bar;
  // No boom today?
  // Boom tomorrow?
  ```

Static Things Bad

- // asm.h
  static inline int clever (int a)
  { return …;}
  - Ok so far
- // user.hh
  #include <asm.h>
  inline void wrapper (int a)
  { clever (a); }
- // bill.cc
  #include “user.hh”
  … wrapper (5); …
- // bob.cc
  #include “user.hh”
  … wrapper (6); …

C++ ODR violation!

- C & C++ have different semantics for ‘inline’

Inline Static

- Header units preserve the existing brokenness
- Implementation defined:
  - Either each import sees a different inline static,
  - Or all imports see the same inline static

GCC Implementation

- Unnamed enums
  - Detected
  - Checked (as of last week!)
- Static Inline
  - Some hacks implemented

Building

- Header-units must be compiled before being imported
- We only find header-units by compiling their importers
- How do users know which header-files can be header-units?