WG21 Number: P0983R0; Date: 2018-04-01; Audience: All; Reply-To: gdr@microsoft.com, richardsmith@google.com

Plan of Record for Making C++ Modules Available in C++ Standards

Authors: Gabriel Dos Reis, Richard Smith

Abstract

This document presents our plan of best effort to make C++ Modules facilities, in part or full, available to C++ programmers via C++ Standards and Technical Specification starting with C++20.

Background

There is a significant overlap between the contents of the Module TS and the model presented at the Jacksonville meeting (P0947R1, hereafter "the Atom proposal"). This is an expression of our intent to make those facilities available to C++ programmers as early as possible, while working on resolving the parts that need more understanding. This is, of course, barring any unforeseen major event.

Plan

Rapperswil meeting (Summer 2018)

We intend to propose doing the following, as an atomic operation:

1. Merge into the IS draft wording supporting the intersection of programs accepted by the Modules TS and the Atom proposal. Specifically, this is the part of the Module TS that supports translation units that start with module declarations, without the ability to export macros from modules nor to #include arbitrary header files into the global module. Such a translation unit could still start with preprocessor directives as long as they don't generate any tokens. For the cases where the Atom proposal and the Modules TS propose different semantics (in particular, the "reachability" rule and the "path of instantiation" rule) we will aim for an approach consistent with EWG's direction to attempt to merge the Atom proposal into the Modules TS. As discussed at Jacksonville, we will determine how to resolve the question concerning dependent ADL (P0923R0) in conjunction with Nathan Sidwell and John Spicer.

2. Merge into the IS draft wording supporting module partitions based on the design described in P0775 (as reviewed with future work encouraged at Rapperswil) and incorporated into the Atom proposal. Some specifics here still need to be worked out, including the impact on build tools and the adoption story.

3. Amend the Modules TS to

    - in addition to the form permitted by the IS, allow a source file that would be a module unit to have the following structure:
        - the first two tokens are module followed by semicolon, as proposed in P0713R1 and adopted by EWG at the Spring 2018 Jacksonville meeting
        - a sequence of preprocessor directives
        - a module declaration, followed by top-level declarations
    - retain the current "reachability" rule for declarations that appear before the module-declaration (where there is no reordering issue)
    - add support for header imports (including importation of macros, but only when importing a legacy header)

Ability to export macros from a named module remains a separate topic from the effort in this stage.

Examples:

1. Planned for the IS (Rapperswil)

```
#include <warnings.h>   // does not generate tokens
export module Algebra.Structures;
export template<typename T>
export concept Ring = requires(T a, T b) { /* ... */ };
export template<Ring T>
struct Matrix { /* .... */ };
```

2. Planned for the TS (Rapperswil)

```
module;
#include <warnings.h>
#if defined WIN32_TARGET
#  include <windows.h>
#elif POSIX_TARGET
#  include <unistd.h>
#endif
export module Fancy.Lib;
import Algebra.Structure;
import "tokens.h";
export struct Window { /* ... */ };
```

Following meetings

We will be working on fleshing out the semantics of header imports, their impacts, and migration recommendations through the TS. The intent is to identify parts of the TS, as informed by feedback and experimentation, that are ready to be merged into C++20+ standards.