Document Number: P0714
Date: 2017-06-06
To: SC22/WG21 CWG/EWG
Reply to: Nathan Sidwell
nathan@acm.org
Re: Working Draft, Extensions to C ++ for Modules, n4647

Identically Named Namespaces and non-Exported Symbols

Nathan Sidwell

This discusses how namespaces and identically-named non-namespace entities interact.

1 Background

The Modules proposal is not entirely clear about how namespaces and identically-named non-namespace entities interact. This has led to confusion in at least this implementor's mind, and in questioning others, some assumptions have become apparent. Those assumptions do not appear supported by the wording of the proposal.

2 Examples

The example I shall use is of two independent modules Foo and Bar containing various exported and non-exported definitions. The question being asked is whether Foo and Bar can be imported into a third module, Quux.

2.1 Interface Namepace and Non-Exported Symbol

Consider module Foo declaring namespace Toto in its interface and module Bar declaring non-exported function Toto in its interface:

```
// Foo interface
export module Foo;
namespace Toto {
}

// Bar interface
export module Bar;
void Toto ();
```

2.2 Implementation Namespace and Non-Exported Symbol

Consider module Foo declaring namespace Toto in its implementation and module Bar declaring non-exported function Toto in its interface:

```
// Foo implementation
module Foo;
namespace Toto {
}
```

```
// Bar interface
export module Bar;
void Toto ();
```

3 Discussion

[7.1]/1 is modified to add:

> A namespace with external linkage is always exported regardless of whether any of its namespace-definition is introduced by export. [Note: There is no way to define a namespace with module linkage.— end note]

This suggests there is no practical difference between the two examples – although namespace `Toto` is only present in the second example's implementation, it doesn't have module linkage and is exported. (However, it is plausible that a problem might only be detected at the final link rather than at module import.)

An assumption that may be being made is that `Foo`'s declaration of namespace `Toto` pervades the entirety of the program. That includes translation units that neither mention it nor import modules that mention it in their interfaces.

However, that does mean that importing both `Foo` and `Bar` (directly or indirectly) is ill-formed, which might be surprising (see non-module equivalent below).

The question reduces to whether a declaration of a namespace in any translation unit creates an empty namespace partition in all translation units, or does a namespace need to be explicitly declared in a translation unit to create a (possibly empty) partition? Is there a difference between the empty set and the set not existing?

3.1 Mailing List Discussion

A brief discussion on the modules mailing list has led to agreement that the example should be ill-formed. One cannot mix namespace and non-namespace entities of the same name in a single program.

Non-Module Behaviour

It has also been pointed out on the modules mailing list that this is not a new module-specific situation. One cannot mix non-namespace external-linkage entities with namespaces of the same name. However, a common behavior of implementations is that such mixing works in practice as the namespace has no symbol associated with it in any object file, and, when the other entity is a function, it will have additional mangling so that it has a distinct link-level symbol.

3.2 Compiler Implementation

Namespaces are containers, with each module observing a different partition of a particular namespace. However namespace members are a little different. A natural representation of a namespace member, at least to this implementor, is for all partitions that it is a member of to refer to the same child namespace object. If the above example is well-formed, name lookup that finds a child namespace in one partition cannot immediately stop safe in the knowledge that there can be no non-namespace bindings of the symbol in other partitions.

4 Conclusion

The modules proposal should be explicit that the example is ill-formed, and indicate this is not a new restriction.