Document Number: P0787r0
Date: 2017-10-13
To: SC22/WG21 CWG/EWG
Reply to: Nathan Sidwell
nathan@acm.org / nathans@fb.com
Re: Working Draft, Extensions to C ++ for Modules, n4681

Proclaimed Ownership Declarations

Nathan Sidwell

The current wording of n4681 specifies that proclaimed ownership declarations are a top-level grammar construct. And not much else.

1 Background

A proclaimed ownership declaration has the grammatical form:

toplevel-declaration
module-declaration
proclaimed-ownership-declaration
declaration

proclaimed-owernship[sic]*-declaration*
`extern` *module module-name* `:` *declaration*
[basic.link,6.5]

It has the following semantics:

1 A *proclaimed-ownership-declaration* asserts that the entities introduced by the declaration are exported by the nominated module. It shall not be a defining declaration.

2 The program is ill-formed, no diagnostic required, if the owning module in the *proclaimed-ownership-declaration* does not export the entities introduced by the declaration.
[dcl.module.proclaim,10.7.4]

1.1 Grammar

The specified grammar only permits such declarations at the global namespace. How is one expected to proclaim ownership within some other namespace? Is it permitted to use a qualified name here? That seems contrary to existing rules of only permitting the introducing declaration of an entity to use an unqualified name.

As noted in p0774r0, a *proclaimed-ownership-declaration* is one of the few uses of the '`module`' keyword. Now that exporting an imported module no longer uses '`export module NAME;`', the use of '`module`' here should be reviewed. Should it be changed, making '`module`' a context sensitive keyword is possible. New keywords always have a risk, and there are codebases using '`module`' as a name in their external APIs.

1.2 Semantics

The intent of the *proclaimed-ownership-declaration* appears to be the module equivalent of a regular `extern` declaration. Namely that some other translation unit is providing a definition of the named entity.

It is not clear why this is needed – for what reason does a translation unit not simply import the named module? Examples would help. It does appear to be a mechanism whereby one sub-module can forward-declare entities in a sibling sub-module. This may be a problem better addressed by module partitions, described in p0775r0.

There are no specified restrictions on the declaration, other than it must be non-defining. That will prohibit function, class and enumeration definitions. It leaves some other declarations unspecified. Are the following permissible?

```
extern module foo : typedef int widget;
extern module foo : using ns::frob;
extern module foo : using namespace t = thing;
extern module foo : static_assert (6);
extern module foo : ;
```

My suspicion is that all but the first are intended to be ill-formed (although exporting *typedefs* and *alias-declarations* have their own problems of not having linkage).

2 Proposal

If there is no good reason for *proclaimed-ownership-declaration*s, it should be deleted.

The remainder of this paper is predicated on the assumption that they are necessary.

2.1 Grammar

I propose moving the *proclaimed-ownership-declaration* into that of a declaration. This will permit proclaiming ownership of non-global-namespace entities.

I further propose the syntax not use '`module`'. We're effectively selectively importing something, so the `import` keyword seems appropriate. This is similar to other languages that use an import keyword to allow both modular import and selective import. For instance Modula-2:

```
DEFINITION MODULE foo ;
IMPORT baz ;
FROM bar IMPORT thing ;
```

Given that a proclaimed-ownership-declaration is the module equivalent of a regular `extern`, perhaps intent would be clearer if the `extern` keyword was present in the *declaration.*

`import` *module-name* `: extern` *declaration* `;`

In conjunction with p0774r0, I propose making '`module`' a context-sensitive keyword.

2.2 Semantics

The declarations introduced by a *proclaimed-ownership-declaration* shall be functions, variables or types (including non-defining templates thereof). They must not be using declarations or directives.

It should be made clear that it is well-formed should the named module be imported (directly or indirectly) either before or after the *proclaimed-ownership-declaration*.

The export description should make clear that a *proclaimed-ownership-declaration* cannot be exported and nor may the entities it declares.

3 Changes to Modules-TS Draft

Remove *proclaimed-ownership-declaration* from the grammar changes in [basic.link,6.5]:

toplevel-declaration:
module-declaration
~~*proclaimed-ownership-declaration*~~
declaration

module-declaration:
`export`$_{opt}$ `module` *module-name attribute-specifier-seq*$_{opt}$ `;`

~~*proclaimed-ownership-declaration*~~
~~`extern module` *module-name* `;` *declaration*~~

Should p0774 be accepted with the 'explicit global module' syntax, the [basic.link,6.5] changes are as follows:

translation-unit:
module-preamble$_{opt}$
~~*toplevel-*~~*declaration-seq*$_{opt}$

~~*toplevel-declaration-seq*~~
~~*toplevel-declaration*~~
~~*toplevel-declaration-seq toplevel-declaration*~~

~~*toplevel-declaration*~~
~~*module-declaration*~~
~~*proclaimed-ownership-declaration*~~
~~*declaration*~~

module-preamble:
module-declaration global-module-declaration$_{opt}$

module-declaration:
`export`$_{opt}$ `module` *module-name attribute-specifier-seq*$_{opt}$ `;`

global-module-declaration:
`module { declaration-seq`$_{opt}$ `}`

~~*proclaimed-owernship-declaration*~~
~~`extern` *module module-name ; declaration*~~

module-name:
module-name-qualifier-seq$_{opt}$ *identifier*

module-name-qualifier-seq:
~~*module-name-qualifier .*~~
module-name-qualifier-seq$_{opt}$ *identifier* `.`

~~*module-name-qualifier*~~
~~*identifier*~~

Should p0774 also be accepted, modify [lex.key,5.11]:

> In 5.11, add the~~se two~~**following** keyword~~s~~ to Table 3 in paragraph 5.11/1: ~~`module` and~~ `import`.

Also, dependent on p0774, document that the `module` should be added as an identifier with special meaning to Table 4 in [lex.name,5.10]/2.

Document that the note in [lex.key,5.11/1 should be modified:[1]

1 An orthogonal correction noticed during editing.

[*Note:* The ~~`export` and~~ `register` keyword~~s are~~ **is** unused but ~~are~~ reserved for future use. — *end note*]

Adjust the changes to *declaration* grammar in [dcl.dcl,10]/1:

declaration:
block-declaration
nodeclspec-function-declaration
function-definition
template-declaration
explicit-instantiation
explicit-specialization
linkage-specification
namespace-definition
empty-declaration
attribute-declaration
export-declaration
module-import-declaration
proclaimed-ownership-declaration

export-declaration:
`export` *declaration*
`export {` *declaration-seq*$_{opt}$ `}`

module-import-declaration:
`import` *module-name attribute-specifier-seq*$_{opt}$ `;`

***proclaimed-ownership-declaration*:**
`import` *module-name* `:` `extern` *declaration*

As an editorial note, it might be worth considering moving the grammars for *export-declaration*, *module-import-declaration* and *proclaimed-ownership-declaration* to their respective defining paragraphs.

Modify [dcl.module.proclaimed,10.7.4]:

1 A *proclaimed-ownership-declaration* asserts that the entities introduced by the *declaration* are exported by the nominated module **with the kind & type specified**. It **may only occur at namespace scope. The declaration may only be an *alias-declaration*, non-defining *simple-declaration* or non-defining *template-declaration*** ~~shall not be a defining declaration~~.

2 **The named module may be explicitly imported (directly or indirectly) before or after the *proclaimed-ownership-declaration*.** The program is ill-formed, no diagnostic required, if the owning module in the *proclaimed-ownership-declaration* does not export ~~the~~ entities **with the kinds and types** introduced by the *declaration*. **A proclaimed-ownership-declaration may not be exported.**