Document Number: p1138r0
Date: 2018-06-22
Reply-to: Aaron Ballman <aaron@aaronballman.com>
Author: Aaron Ballman <aaron@aaronballman.com>
Audience: SG1, Evolution Working Group

Deprecating `ATOMIC_VAR_INIT`

Motivation

The `ATOMIC_VAR_INIT` macro was introduced as a way to constant initialize atomic variables in a manner suitable for both C and C++ source code. It was believed that this was necessary in order to allow an atomic initialization to also initialize embedded locks, as it provides a syntactic marker for the compiler to trivially know where to generate the embedded lock initialization. However, to date, no implementation has required this latitude in the form of an explicit syntactic marker. Further, exposing the initialization via a macro results in the macro being unsuitable for initializing certain types, like aggregates. For instance,

```
#include <atomic>

struct S {
  int i, j;
};

std::atomic<S> s1 = ATOMIC_VAR_INIT({});
std::atomic<S> s2 = ATOMIC_VAR_INIT({1});
std::atomic<S> s3 = ATOMIC_VAR_INIT({1, 2});
```

In this code example, the initialization of `s3` is ill-formed in all implementations I was able to test (Clang, GCC, MSVC, and ICC). The typical implementation of `ATOMIC_VAR_INIT` looks like:

```
#define ATOMIC_VAR_INIT(value) (value)
```

Because the `ATOMIC_VAR_INIT` macro only accepts a single argument, the presence of the comma in what the programmer believes to be an aggregate initializer is instead interpreted as a comma separating function-like macro arguments.

WG14 considered these concerns in DR 485 and elected to deprecate the `ATOMIC_VAR_INIT` macro in C17 with the intention of removing the macro in a future revision of the C standard [DR 485]. Use of the `ATOMIC_VAR_INIT` macro is no longer required when explicitly initializing an atomic object in C; instead, automatic variables are left in an indeterminate state and the zero-initialization of a static or thread-local variable is required to leave the object in a valid state.

This paper is intended to realign C and C++ by deprecating the `ATOMIC_VAR_INIT` macro. Given the overlap with P0883 regarding atomic initialization, which was approved by SG1 in Rapperswil 2018 to target LEWG for C++2a, this paper does not propose modifying the behavior of default initialization of atomic objects, retaining the C++ status quo for that behavior [P0883]. If P0883 is not adopted but this

proposal is, the result will be an inconsistency where a variable with static or thread-local storage duration will be uninitialized in C++ but zero-initialized in C.

Proposed Wording

Delete [atomics.types.operations]p4 (the definition of ATOMIC_VAR_INIT).

Add new Subclause to Annex D:

D.XX Deprecated atomic operations [depr.atomics]

1 The following macro is declared in addition to those operations defined in 32.6.1:

```
#define ATOMIC_VAR_INIT(value) see below
```

The macro expands to a token sequence suitable for constant initialization of an atomic variable of static storage duration of a type that is initialization-compatible with value. [Note: This operation may need to initialize locks. — end note] Concurrent access to the variable being initialized, even via an atomic operation, constitutes a data race. [Example:

```
atomic<int> v = ATOMIC_VAR_INIT(5);
```

— end example]

Acknowledgements

Thank you to Hans Boehm for reviewing the paper.

References

[DR 485]
Clarification Request Summary for C2x Version 1.14. Blaine Garst. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2257.htm#dr_485

[P0883]
Fixing Atomic Initialization, Rev0. Nicolai Josuttis. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0883r0.pdf