Atomic C1x/C++0x Compatibility Refinements

ISO/IEC JTC1 SC22 WG14 Document N1526

Blaine Garst, blaine@apple.com
Oct 14, 2010

Introduction

(This document supercedes N1522 entirely)

The adoption of the Atomic Proposal N1485 by WG14 leads to an opportunity for further simplification of the C1x draft by eliminating section 7.17.6 which defines a limited set of opaque structures containing atomic elements. These elements were also correspondingly defined in C++0x as interoperable data types. Prior to N1485 these types served as the exclusive set of atomic types defined by C1x, but are now unnecessary due to the introduction of N1485's productive syntax for atomics.

This paper, along with Lawrence Crowl's C++0x paper N3164, propose to simplify both standards by eliminating these specific data types.

Additionally, an oversight is corrected. Let's start there.

_Atomic can mix with const

The first formulations of the atomic proposal were deliberately bare-boned with the primary intent of proving useful declarative syntax to enable compound assignment operators to have an intuitive and useful meaning. From that perspective, a **`const _Atomic`** made no sense. As the proposal evolved to what was accepted in Colorado, the need for **`const _Atomic`** was overlooked. The C1x draft standard N1516 itself contains in 7.17.5.1

`_Bool atomic_is_lock_free(`*atomic_type* **`const volatile *obj);`**

which, in general, indicates that no updates are permissible to such a qualified parameter. Also, of course, C++0x allows const atomic and so this should be seen as an oversight that, left uncorrected, would undermine C++0x and C1x compatibility.

Remedy:

Section 6.2.5 Types, paragraph 27

change "which may combine with **volatile** and **restrict**." to "which may combine with **volatile**, **const**, and **restrict**."

Eliminate Definitions and References to most atomic_xyz types

Of the many atomic_xyz types defined in the current draft, only **atomic_flag** need remain. All others have natural _Atomic qualified definitions. To that end

Section 7.17.1 Introduction,

In paragraph 4 remove the mention of **atomic_bool** and **atomic_address**.

In paragraph 5 remove the sentence "The **atomic_address** atomic type corresponds to the **void *** non-atomic type."

Also change "For atomic address types" to "For atomic pointer types".

Section 7.17.2.1 The ATOMIC_VAR_INIT macro

In paragraph 4 Example change

```
atomic_int guide = ATOMIC_VAR_INIT(42);
```

to

```
_Atomic int guide = ATOMIC_VAR_INIT(42);
```

Section 7.17.2.2 The atomic_init generic function

In paragraph 5 Example change

```
atomic_int guide;
```

to

```
_Atomic int guide;
```

Remove section 7.17.6

Paragraph 1 defines what were to be compatibility structures with C++0x - these are no longer needed.
Paragraph 2 is not necessary, section 7.17.7 is self explanatory.
Paragraph 3 and 4 are not necessary because atomic_bool and atomic_address are no longer referenced in 7.17.1
Paragraph 5 is not necessary because the representation difference is mentioned in 6.25 Types Paragraph 26

Section 7.17.7.5 The atomic_fetch and modify generic functions

In paragraph 1 remove the sentences "Only addition and subtraction are applicable to **atomic_address**. None of these operations is applicable to **atomic_bool**."

Appendix B.16

Remove all **atomic_***integral* references other than **atomic_flag**. More precisely, remove

atomic_uint
atomic_long
atomic_ulong
atomic_llong
atomic_ullong
atomic_char16_t
atomic_char32_t
atomic_wchar_t
atomic_int_least8_t
atomic_uint_least8_t
atomic_int_least16_t
atomic_uint_least16_t
atomic_int_least32_t
atomic_uint_least32_t
atomic_int_least64_t
atomic_bool
atomic_address
atomic_char
atomic_schar
atomic_uchar
atomic_short
atomic_ushort
atomic_int
atomic_int_fast8_t
atomic_uint_fast8_t
atomic_int_fast16_t
atomic_uint_fast16_t
atomic_int_fast32_t
atomic_uint_fast32_t
atomic_int_fast64_t
atomic_uint_fast64_t
atomic_intptr_t
atomic_uintptr_t
atomic_size_t

atomic_intmax_t
atomic_ptrdiff_t
atomic_uintmax_t