OCT 11, 2010

Draft 2

ATOMIC REFINEMENTS, N1522

BLAINE GARST

APPLE INC.

blaine@apple.com

1 Introduction

The adoption of the Atomics N1485 proposal by WG14 leads to an opportunity for further simplification of the C1x draft by eliminating section 7.17.6 which defines opaque structures containing atomic elements. These were necessary prior to N1485 as the exclusive set of atomic types defined by C1x, but are now unnecessary.

The C++0x concurrency working group is in agreement that this section is no longer necessary in C1x (see n1508) and that corresponding changes to C++0x should be made.

2 Improve the introduction to atomics

Section 7.17.1 Introduction

Paragraph 4

replace "which is [sic] structure type" and "which is a structure type"

with "which is a type"

for the descriptions of atomic_flag, atomic_bool, and atomic_address.

Replace "the atomic analog of a pointer type; and several atomic analogs of integer types."

with "the atomic void * type."

By defining atomic_address as _Atomic void * we implicitly define the unit of addition on this type to be that of one byte which is otherwise called out in section 7.17.6.

3 Replace uses of *atomic_int* with *_Atomic int* in two examples.

Section 7.17.2.1 The ATOMIC_VAR_INIT macro

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

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 already defined in 7.17.1

Paragraph 5 is not necessary because the representation difference is mentioned in 6.25 Types Paragraph 26

4 _Atomic can mix with const

An oversight in N1485 is that _Atomic is not specified to mix with const, though it does with volatile and restrict. C++0x allows const atomics, and 7.17.5.1 shows the use of const as a qualifier in use as a parameter modification restriction for atomic_is_lock_free.

Section 6.25 Types

Paragraph 27

change “may combine with volatile and restrict.” to “may combine with volatile, const, and restrict.”