ISO/IEC JTC 1/SC 22/WG 14: N1272
Author: Nick Stoughton

C - POSIX Liaison

This paper describes the issues arising from the most recent WG 14 C meeting that affect POSIX.

Thread Proposals for C++

There were at least three proposals on the table for how to procede with threading in the C revision:

1. Just do the underlying mechanisms for multiple threads (memory model, atomic data types, thread local storage, sequence points, etc), but do not introduce any Thread Launching API. [proposal in N1257].

2. Add the pthread* interfaces from POSIX. There is a question on how much of the pthread interface is appropriate, but The Austin Group has agreed to assist in developing the words for C. [proposal in N1257]

3. Add a slightly higher abstraction level, based on the existing practice of Dinkumware, which should also be compatible with whatever C++ does, that would be a thin veneer over any underlying pthread* and windows threading API.

Each of these proposals has strong arguments in favor of it, and somewhat weaker arguments against it. There is widespread existing practice in this space (mainly option 2, but to a lesser extent option 3 (Dinkumware)).

The entire committee was in favor of adding the underlying mechanisms, and it
is almost certain that the C++ memory model, atomics etc will be included in
the revision of C.

However, when it came to deciding beyond that there was considerably less consensus.

Option 1: 5 in favor, 10 opposed, 3 abstain.

Option 2: 5 in favor, 7 opposed, 4 abstain.

Option 3: 15 in favor, 2 opposed, 1 abstain.

As a result, an Action Item was given to Bill Plauger to develop a paper describing option 3 for consideration at the next meeting. Neither of the other two options is specifically off the table, but the "thin-layer over the OS specific one" seems to have the most traction, providing greater portability for applications that use this layer. It was observed that the Dinkumware library was approximately 1500 lines of code + 1500 lines of header for *both* the pthread and windows implementations (which are ifdefed together).

The committee sees its job as harmonizing competing existing practice, and believes that there is a useful intersection that can be implemented with low overhead, and at this point has requested a more detailed proposal.

CERT Secure Programming Document
================================
The committee spent a short while discussing the CERT C Programming Language Secure Coding Guidelines in N1255. In the past, other groups have written documents similar to this (and often not as good as this), and published them without asking for committee input. This time at least that step has been taken. The document contains a section on POSIX, and the Austin Group has been invited to collectively review this section. If the Austin Group does nothing, it is possible that such silence will be taken as assent, so WG 14 would strongly encourage Austin Group members to spend some time looking over this.