From: Frank Farance
Organization: Farance Inc.
Telephone: +1 212 486 4700
Fax: +1 212 759 1605
E-mail: frank@farance.com
Date: 1995-08-25
Document Number: WG14/N463 X3J11/95-064
Subject: Impact of adding WG11's LIA-1, LID, and LIPC features.

---

1. OVERVIEW

The paper investigates the possibility of providing C bindings to the following language independent standards:

- ISO 10967-1, ''Information technology -- Language independent arithmetic -- Part 1: Integer and floating point arithmetic'' (LIA-1).

- ISO DIS 11404, ''Information technology -- Language independent datatypes'' (LID).

- ISO DIS 13886, ''Information technology -- Language independent procedure calling'' (LIPC).

This paper explores the goals, purpose, and feasibility of providing these features in C9X.

2. BACKGROUND INFORMATION

These standards were developed by ISO JTC1/SC22/WG11. The U.S. TAG is X3T2.

2.1 LIA-1

X3J11 and X3J11.1 had received several presentations on the LIA-1 work as it progressed over the years. Originally, X3J11 and X3J11.1 had little interest in LIA-1 because:

- The paradigm of language independent arithmetic was inappropriate for C.

- The C binding would require substantial changes to many C implementations, e.g., requiring exceptions on integer overflow.

- The C binding wouldn't be in the ``Spirit of C'' because the loss of performance was significant.

- X3J11 and WG14's interests weren't well represented in WG11.

Since then, the LIA-1 has changed in favor of an approach that is workable among many languages. I believe part of this success has been developing and providing sample bindings of LIA-1 to common languages (including C).

2.2 LID

The LID draft standard has not been presented, to my knowledge, to X3J11 or WG14. X3J11 and WG14 have not been asked to review the DIS nor have they received copies from X3T2 or WG11.

2.3 LIPC

Recently X3J11 has been asked to review the LIPC DIS. To my knowledge, WG11 has not asked WG14 for its review. The X3J11 review has been distributed in the current mailing for WG14.

2.4 Other Standards

The LIPC DIS refers to the RPC standard (ISO 11578) as another standard with similar, but slightly divergent capabilities. To my knowledge, X3J11 or WG14 has not reviewed the RPC standard.

The IEEE Std 1596.5, ``Shared-data formats optimized for scalable coherent interface (SCI) processors'' has been reviewed informally by WG14 with respect to extended integer range features.

2.5 C9X Activities

WG14 has several extended integer (SBEIR, "inttypes.h", BIGINT, OAX, REP) that are related to the LID integer datatype.

WG14 has a floating point extensions (FPCE) proposal that is related to the LID real datatype. The FPCE proposal also intersects with the LIA-1 standard.

WG14 has several complex arithmetic proposals that are related to the LID complex datatype.

WG14 has a proposal for adding a boolean datatype, similar to the LID datatype.

494

WG14 has a proposal for adding class-like features to C9X. Although many of the LID datatypes could be implemented via a class mechanism, the missing component in C9X (and C++) is a convenient mechanism for extending the promotion rules. In C++, this is implemented by operator overloading (itemized each promotion). The C++ approach is impractical for adding these promotions because either all the promotions must be itemized (1000's of them) or the implementation promotes to the largest type (a performance hog). Thus, the C++ approach is not within the ''Spirit of C''.

WG14 has had some discussion on characters, extended characters, extended identifiers, wide characters, and so on. This discussion relates to the LID character datatype.

3. SUMMARY OF FEATURES

3.1 LIA-1

This standard specifies characteristics of arithmetic operations that are language independent. This standard does not specify the operands of the operators. Each language that binds to LIA-1 must provide a mechanism for accessing certain operators and the documentation describing certain characteristics of these operations.

This standard provides a sample C binding in its annexes. The WG14 document N461 ''C Binding of ISO 10967-1 (LIA-1)'' proposes to incorporate this feature into C9X.

In summary, providing an LIA-1 binding to C is relatively easy because it adds a header, a couple functions, and a requirement for documenting the arithmetic operators and features. LIA-1 doesn't change the behavior of C programs.

3.2 LID

This DIS specifies several datatype and datatype generation mechanisms. Each is described by its domain (a set of acceptable values), its operational properties (e.g., integers are: ordered, each, numeric, unbounded), and its characterizing operations (methods in object-oriented terminology).

The following datatypes are specified in LID:

Primitive datatypes: boolean, state, enumerated, character, ordinal, date-and-time, integer, rational, scaled, real, complex, void.

Subtypes: range, selecting, excluding, extended, size, explicit subtypes.

Generated datatypes: choice, pointer, procedure.

495

Aggregate datatypes: record, set, bag, sequence, array, table.

Defined datatypes.

One can easily imagine extending C to include these datatypes (natively) or building class libraries to define them. Unfortunately, the typing paradigm in the DIS misses one important point: C programmers choose a type not only because of its functionality but also because of its implementation attributes. For example, both "char" and "int" provide integers in the range of 0 to 127, but they have different implementation characteristics: "char" might require less storage and "int" might provide faster access. Although the C Standard does not require these implementation features, choosing the appropriate *implementation* of a type is a significant aspect of most C programs.

If LID had considered several implementation variations, it would have included a discussion of promotion rules. However, LID requires a single mapping for each datatype. The following is from subclause 11.2:

> For each LI datatype (primitive or generates), the mapping shall specify whether the LI datatype is supported by the language (as specified in 11.4), and if so, identify a single corresponding internal [i.e., C] datatype.

Thus, a LID datatype such as ``integers in the range of 0 to 127'' must map into, say, "char" or "int" but not both. This is impractical for C programming.

Integers aren't the only example of this problem. Booleans can be implemented as a 1-bit bit field (smallest, not addressable), a "char" (smallest, addressable), an "int" (addressable, possibly not the smallest, possibly faster than "char"). Similarly the date-and-time type could be implemented as a "time_t" or an ISO 8601 date-time character array. LID would require its binding to choose one or the other.

While it may be possible to provide a C binding to LID, it won't be practical because the binding will include either the largest ranges (widest applicability, lowest performance), or the smallest ranges (smallest applicability, fastest performance), or some average ranges (not widely applicable, not great performance). Regardless of which compromise is chosen, there probably won't be widespread use because each binding has its own set of significant disadvantages.

In summary, the LID DIS has some good ideas, but appears to be more of an academic approach without recognition of implementation constraints. The LID DIS would benefit by providing sample bindings to C and other common languages. The sample Pascal binding is convenient for presenting LID features, but is not applicable for C (or, say, Fortran) bindings.

3.3 LIPC

This DIS specifies a generic model for procedure calling conventions. The model is based upon a ''client'' calling a ''server''. The model seems applicable to many programming languages and remote procedure call systems.

The LIPC model specifies a single procedure calling convention rather than a suite of calling conventions. Rather than the programmer choosing lightweight procedure call (for same process calls) or a heavyweight procedure call (for remote procedure calls) as necessary, the LIPC model requires that there be a single implementation of the language binding. Thus, the programmer gets no choice of implementations. The lightweight implementation performs well, but doesn't handle all the exceptions (e.g., network failure). The heavyweight implementation is robust, but performs poorly and requires much more programming overhead. Neither approach is the right one, yet the LIPC binding must choose one.

In summary, a C binding of LIPC won't be widely used for the same reasons of LID: the paradigm does not acknowledge varying implementations and their attributes. See WG14 document N462 ''X3J11 Review of ISO DIS 13886'' for more details of the LIPC DIS. Like LID, LIPC could benefit by providing sample bindings to C.

4. CONCLUSIONS

The LIA-1 standard should be included in C9X. This binding is low cost for implementations and doesn't affect C programs.

The LID paradigm is infeasible for C9X. Given the timetable of C9X, it is unlikely that LID will be improved before the cutoff for new proposals. I recommend that we postpone the development of a C binding to LID until we see a binding more hospitable to the C type system and existing programming styles.

Some of the LIPC problems could be addressed within the timetable of C9X. Unfortunately, LIPC is dependent upon LID datatypes for its interface definition, so even if LIPC were fixed, LIPC couldn't be included until LID was fixed. Thus, I recommend against the inclusion of LIPC until these problems have been solved.

WG14 and X3J11 should have better communication with WG11 and X3T2. It is unfortunate that the LID and LIPC efforts have reached DIS will little input from the C community. Since C and its variants are probably the most widely used programming languages for systems programming, WG11 and X3T2 should be more sympathetic to C needs.