WG14/N~~444~~(X3J11/95-~~045~~)
Rex Jaeschke, Chair X3J11
rex@aussie.com
August 2nd, 1995

The C9X Charter

1 Introduction

At the WG14/X3J11 meeting in Kona, Hawaii, in December 1993, there was general agreement the committee should start thinking about the next version of the C Standard. I accepted an action item to draft a revision charter, the result of which is this paper. The intention of this paper is to present a statement of principles and a plan of attack. It does not identify any technical issues since those are immaterial at this stage.

Although the committee is not yet required to begin work on a revised standard, there is much happening that can or does influence C directly. Examples are the evolution of C++ (and object-oriented programming in general), the numerical extensions being proposed by X3J11, internationalization and advancements in character set standardization, and cross-language standards and bindings.

2 Original Principles

Before embarking on a revision of the C Standard, it is useful to reflect on the charter of the original drafting committee. According to the original Rationale Document in the section entitled "Purpose:"

The work of the Committee was in large part a balancing act. The Committee has tried to improve portability while retaining the definition of certain features of C as machine-dependent. It attempted to incorporate valuable new ideas without disrupting the basic structure and fabric of the language. It tried to develop a clear and consistent language without invalidating existing programs. All of the goals were important and each decision was weighed in the light of sometimes contradictory requirements in an attempt to reach a workable compromise.

In specifying a standard language, the Committee used several guiding principles, the most important of which are:

1. **Existing code is important, existing implementations are not.** A large body of C code exists of considerable commercial value. Every attempt has been made to ensure that the bulk of this code will be acceptable to any implementation conforming to the Standard. The Committee did not want to force most programmers to modify their C programs just to have them accepted by a conforming translator.

   On the other hand, no one implementation was held up as the exemplar by which to define C: It is assumed that all existing implementations must change somewhat to conform to the Standard.

2. **C code can be portable.** Although the C language was originally born with the UNIX operating system on the DEC PDP-11, it has since been implemented on a wide variety of computers and operating systems. It has also seen considerable use in cross-compilation of code for embedded systems to be executed in a free-standing environment. The Committee has attempted to specify the language and the library to be as widely implementable as possible, while recognizing that a system must meet certain minimum criteria to be considered a viable host or target for the language.

3. **C code can be non-portable.** Although it strove to give programmers the opportunity to write truly portable programs, the Committee did not want to *force* programmers into writing portably, to preclude the use of C as a "high-level assembler;" the ability to write machine-specific code is one of the strengths of C. It is this principle which largely motivates drawing the distinction between *strictly conforming program* and *conforming program*.

4. **Avoid "quiet changes."** Any change to widespread practice altering the meaning of existing code causes problems. Changes that cause code to be so ill-formed as to require diagnostic messages are at least easy to detect. As much as seemed possible, consistent with its other goals, the Committee has avoided changes that quietly alter one valid program to another with different semantics, that cause a working program to work differently without notice. In important places where this principle is violated, the Rationale points out a QUIET CHANGE.

5. **A standard is a treaty between implementor and programmer.** Some numerical limits have been added to the Standard to give both implementors and programmers a better understanding of what must be provided by an implementation, of what can be expected and depended upon to exist. These limits are presented as *minimum maxima* (i.e., lower limits placed on the values of upper limits specified by an implementation) with the understanding that any implementor is at liberty to provide higher limits than the Standard mandates. Any program that takes advantage of these more tolerant limits is not strictly conforming, however, since other implementations are at liberty to enforce the mandated limits.

6. **Keep the spirit of C.** The Committee kept as a major goal to preserve the traditional *spirit of C*. There are many facets of the spirit of C, but the essence is a community sentiment of the underlying principles upon which the C language is based. Some of the facets of the spirit of C can be summarized in phrases like

   (a) *Trust the programmer.*

   (b) *Don't prevent the programmer from doing what needs to be done.*

   (c) *Keep the language small and simple.*

   (d) *Provide only one way to do an operation.*

   (e) *Make it fast, even if it is not guaranteed to be portable.*

   The last proverb needs a little explanation. The potential for efficient code generation is one of the most important strengths of C. To help ensure that no code explosion occurs for what appears to be a very simple operation, many operations are defined to be *how the target machine's hardware does it* rather than by a general abstract rule. An example of this willingness to live with *what the machine does* can be seen in the rules that govern the widening of char objects for use in expressions: whether the values of char objects widen to signed or unsigned quantities typically depends on which byte operation is more efficient on the target machine.

   One of the goals of the Committee was to avoid interfering with the ability of translators to generate compact, efficient code. In several cases the Committee has introduced features to improve the possible efficiency of the generated code; for instance, floating point operations may be performed in single-precision if both operands are `float` rather than `double`.

3 Additional Principles

At the WG14 meeting in Tokyo, Japan, in July 1994, the original principles were re-endorsed and the following new ones were added:

7. **Support international programming.** During the initial standardization process, support for internationalization was something of an afterthought. Now that internationalization has proved to be an important topic it should have equal visibility with other topics. As a result, all revision proposals submitted shall be reviewed with regard to their impact on internationalization as well as for other technical merit.

8. **Codify existing practice to address evident deficiencies.** Only those concepts that have some prior art should be accepted. (Prior art may come from implementations of languages other than C.) Unless some proposed new feature addresses an evident deficiency that is actually felt by more than a few C programmers, no new inventions should be entertained.

9. **Minimize incompatibilities with C90 (ISO/IEC 9899:1990).** It should be possible for existing C implementations to gradually migrate to future conformance, rather than requiring a replacement of the environment. It should also be possible for the vast majority of existing conforming C programs to run unchanged.

10. **Minimize incompatibilities with C++.** The committee recognizes the need for a clear and defensible plan with regard to how it intends to address the compatibility issue with C++. The committee endorses the principle of maintaining the largest common subset clearly and from the outset. Such a principle should satisfy the requirement to maximize overlap of the languages while maintaining a distinction between them and allowing them to evolve separately.

    Regarding our relationship with C++, the committee is content to let C++ be the "big" and ambitious language. While some features of C++ may well be embraced, it is not the committee's intention that C become C++.

11. **Maintain conceptual simplicity.** The committee prefers an economy of concepts that do the job. Members should identify the issues and prescribe the minimal amount of machinery that will solve them. The committee recognizes the importance of being able to describe and teach new concepts in a straightforward and concise manner.

4 Important Observations

During the revision process, it will be important to consider the following observations:

1. Regarding the 11 principles, there is a tradeoff between them—none is absolute. However, the more the committee deviates from them, the more rationale needed to explain the deviation.

2. There has been a very positive reception of the standard from both the user and vendor communities.

3. The standard is not considered to be broken. Rather, the revision is needed to track emerging and/or changing technologies and internationalization requirements.

4. Most users of C view it as a general-purpose high-level language. While "higher level" constructs can be added, they should be done so only if they don't contradict the basic principles.

5. There are a good number of useful suggestions to be found from the public comments and defect report processing.

5 Sources of Influence

Areas to which the committee shall look when revising C include:

1. Incorporate Amendment 1.

2. All technical corrigenda and records of response.

3. Future directions in current standard.

4. Features currently labeled obsolescent.

5. Cross-language standards groups work.

6. Requirements resulting from JTC1/SC2 (character sets).

7. The evolution of C++.

8. The evolution of other languages particularly with regard to interlanguage communication issues.

9. Other papers and proposals from member delegations, such as the numerical extensions Technical Report being proposed by X3J11.

10. Other comments from the public at large.

11. Other prior art.

6 Submission Guidelines

Without a set of acceptance criteria, judging any technical proposal becomes a highly subjective, and definitely emotional, exercise. It also wastes a lot of time and energy. Therefore, submittors are encouraged to keep all the guiding principles in mind when making submissions.

Guidelines for the submission of proposals will be provided. Each submission shall contain a cover page containing responses to a number of questions and further summary information enabling the essence of a submission to be distilled simply by reading that cover. The information requested will include such things as: title, author, author affiliation, date, document number, abstract, proposal category (e.g., editorial, correction, new feature), prior art, and target audience. Proposals that are not directly linked must be submitted separately, each with their own document number and cover page.

Submissions *must* be sponsored in the same way as defect reports; that is, either by the convener of WG14, WG14 itself, or by a WG14 national member body. This provides a filtering process and allows submissions to be rejected early in the process if they violate the revision principles. It also allows substantially incomplete or disjoint proposals to be returned for further refinement.

7 Documentation

A rationale document will be maintained and an editor will be appointed.

An editor for the revised standard will be appointed. The initial job of the editor will be to integrate Amendment 1 and technical corrigenda into a single base document against which the committee can work when considering and/or preparing technical papers as well as in handling current and future defect reports.

8 Revision Schedule

The milestones for the revision process are:

1. CD Registration – December 1996
2. CD Ballot – December 1997
3. DIS Ballot – December 1998

This schedule allows for the formal adoption of a revised standard by the end of 1999.

The purpose of this schedule is twofold:

1. To provide the public with a reasonably accurate idea of when a revised C Standard is likely to appear.
2. To keep the revision process focused.