N363 048

WG14/~~N349~~ (X3J11/94-~~034~~)

Rex Jaeschke, Chair X3J11
rex@aussie.com
June 26, 1994

A Charter for Revising Standard C

1 Introduction

At the WG14/X3J11 meetings in Kona, Hawaii, in December 1993, there was general agreement we 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 set of questions and a possible plan of attack. It does not identify any technical issues since I believe those are immaterial at this stage.

Although we are 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 character set standardization advancements, and cross-language standards and bindings.

2 A Bit of History

Before we embark on a revision of the C Standard, it is useful to reflect on the charter of the original drafting committee. According to the original Rational 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.
> 2. C code can be portable.
> 3. C code can be non-portable.
> 4. Avoid "quiet changes."
> 5. A standard is a treaty between implementor and programmer.
> 6. Keep the spirit of C, which 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.*

3 Where To Begin

When the topic of revising C comes up, the discussion very quickly moves to technical details of a feature-oriented nature. We all have our list of favorite things we'd like to add. However, identifying the specific technical issues must be one of the things we defer until sometime later as we prepare for the next revision. Before we jump full speed into a whole host of extensions we need to consider a number of very important issues:

1. How has the standard been accepted by the user community? Do they know or even care about it? How has it been accepted by the implementor community? What can we learn from the validation process? [X3J11: There has been a very positive reception of the standard from both the user and vendor communities.]

2. What are we fixing? Is anything really broken? [X3J11: No!] If so, what? Was it always broken or have circumstances changed that now lead us to a different conclusion? [X3J11: We need to look at advances in areas such as file systems (directory support), multithreading, and numerical programming needs.]

3. How has our original charter held up over time? [X3J11: Very well. Let's use it again this time.]

4. Does the fact that most users of C view it as a general-purpose high-level language change the way we look at it? What shortcomings is C perceived to have? Are they addressed by C++? Should they be addressed by a revised C? [X3J11: We can add "higher level" constructs but only if they don't contradict the original charter.]

5. What can we learn from the interpretation/defect report process? [X3J11: There are a good number of useful suggestions to be found from the public comments.]

6. What can our relationship with the C++ world be? What do we want it to be? What are our obligations regarding compatibility with C++ and C++'s with C? What does the C++ world want us to be? Is the long-term future of C a proper (as proper as possible but no more proper) subset of C++? Can synchronization of C and C++ standardization actually be achieved? If so, how? [X3J11: This is a long-term issue. We certainly should not derail the C++ standardization process by having them get into some synchronization mode with us now. Wait until they have frozen their standard.] What are the implications of a revised C containing features not already part of C++ or that do not readily lend themselves to being added to C++?

7. What other official and ad hoc standardization efforts exist that can influence the future of C? What are our obligations to, and interest in, interfacing with those efforts? What is their time scale? How should we deal with C language binding efforts?

4 Sources of Influence

Areas to which we must look when revising C include:

1. Incorporate the normative addendum.

2. Review all defect reports, technical corrigenda, and records of response. [X3J11: Also public review comments.]

3. Review future directions in current standard.

4. Review the current obsolescence list.

5. Cross-language standards groups work: LIA, POSIX, I18N, bindings such as GKS, and parallel processing. [X3J11: Also X/Open.]

6. The evolution of C++. [X3J11: Also Fortran 9X (for interlanguage issues).]

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

8. Other prior art.

178

5 Some Recommendations

Noticeably absent from the list above is "new inventions." We should think about making a strong case for accepting only those concepts that have some prior art, and that the kind of pure invention that is now happening within X3J16/WG21 be actively discouraged for C. Unless some proposed new feature addresses an evident deficiency that is actually felt by more than a few C programmers, we should entertain no new inventions.

We should state a preference for an economy of concepts that do the job. We should identify the issues and prescribe the minimal amount of machinery that will solve them.

It should be possible for existing C implementations to gradually migrate to future conformance, rather than requiring a replacement of the environment. For example, we should be very careful about changing things such as the current external linkage mechanism.

We need a clear and defensible plan with regard to how we intend to address the compatibility issue with C++. We need to state the principle of 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.

Once we have completed revision of our charter, we should publish at least a tentative schedule, so people have some idea when the next official C Standard is likely to appear. This will also help to keep us focused.

6 Establishing Some Guidelines

We need to establish some reasonably hard and fast rules and priorities. 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.

We also need a way of streamlining proposals. For example, we might establish a list of questions against which each proposal is measured. We could require that suggestions that are not directly related be submitted separately, assigned unique IDs, and have a specific format. Why not require the submittor to get organized and to do as much of the research as possible? We should also provide a way to screen new ideas without requiring a complete proposal.

7 Record Keeping

The worst thing we can do is to not have an organization in place to deal with submissions. I see the need for at least one person to coordinate new submissions, to acknowledge receipt of them, to respond to inquiries from the public, and to reject submissions if they are incomplete format-wise.

We should have someone maintaining a rationale document from the very beginning.

We will need to find a redactor for the revised standard itself. And while there would be no new work to be added to the standard for some time, work should begin early to integrate the normative addendum, defect reports, and the like, so they are incorporated into a single base document against which we can work when considering/preparing technical papers as well as in handling our current defect report load.

8 Handling the Workload

Based on the success of the subgroups used to produce the first standard and to handle defect reports, I suggest we create separate subgroups to deal with environment, language, and library.

We should also have a number of groups/focal points for the topics identified in "Sources of Influence" above. Given the specific nature of some of these topics, not all members will have the background and/or interest in each topic. We will make more progress if people can work on topics in their area of interest and ability.

I expect a good amount of legwork will be done between meetings and that meetings will involve reporting on the status of the subgroups, subgroup meetings, and dealing with issues arising out of the subgroups.