Doc No: X3J16/95-0054 WG21/N0654
Date: 28 Feb 95
Project: Programming Language C++
Ref. Doc: X3J16/93-0046 WG21/N0253
X3J16/94-0015 WG21/N0402
Reply to: Michael J. Vilot
ObjectCraft, Inc.
7 Colby Ct., suite #4–321
Bedford NH USA 03110–6427
mjv@objects.mv.com

Freestanding Implementations

1. Introduction

In October, 1993,[1] Neal Gafter recommended that C++ adopt the C standard's description of hosted and freestanding implementations (93–0046/N0253). This proposal was most likely discussed by the Environment WG at the San Jose meeting, but the present WP does not reflect its wording. The present WP contains an issue note (Box 78, page 17–18) in §17.3.1.3 indicating that a "freestanding implementation" is still an undefined term.

This proposal recommends changes to Clauses 1, 3, and 17 to incorporate Neal's proposal.

2. Discussion

The essential change is to allow an implementation-defined "minimal subset" of C++. The concern is that C++ is too large to be implemented for embedded systems. CD Registration Ballot comments echoed this concern:

> **France**: "The core language shall be separated from the libraries." (R-25)[2]
> **Germany**: "The DIN C++ Working Group is concerned about the size and complexity of the proposed C++ Language and Library." (R-41)
> **Netherlands**: [The C++ Standard] "should contain the definition of the language itself and a minimal set of libraries (effectively only the language support libraries …)." (R-44)
> **New Zealand**: "The absolutely horrendous size of the proposed library is not acceptable. … Once again, it is necessary to ask 'What about the implementation for a 4-bit micro-controller?'" (R-53)

This proposal addresses these concerns by recycling the C Standard's definition of *hosted* and *freestanding* implementations. An C++ implementation for a freestanding environment has to define the semantics of program start & termination, and the available library facilities.

The main advantage of this approach is that it is compatible with C. This should meet the expectations of embedded systems developers moving from C to C++.

3. Proposal

Affects Clauses: 1, 3, 18.

1.8 Program execution **[intro.execution]**

Add something like the following as ¶s 2 and 3:

Two kinds of implementations are defined: *hosted* and *freestanding*. A freestanding implementation is one in which execution may take place without the benefit of an operating system.

---

[1] And again in January, 1994.
[2] These reference numbers correspond to those used by Sam Harbison in Message `c++std-admin-87`.

For a hosted implementation, this Standard defines the semantics of program start, execution, and termination, as well as the set of available libraries. A freestanding implementation has implementation-defined semantics for program start and termination (3.6, Start and termination), and an implementation-defined set of library facilities (17.3.1.3, Freestanding implementations).

3.6 Start and termination [basic.start]

Add something like the following as ¶1:

For freestanding implementations (1.8), the semantics of program start and termination, and the timing of static constructors and destructors, is implementation-defined. The rest of this subclause applies to hosted implementations.

17.3.1.3 Freestanding implementations [lib.freestanding]

Change this subclause to something like the following:

A hosted implementation shall include all of the C++ standard library components described in Clauses 17–27. A freestanding implementation (1.8, 3.6) shall provide an implementation-defined subset of these components. This subset shall include at least the components from Clause 18, Language support library, as described in Table 25:

Table 25—Freestanding implementation minimum library

Subclause	Header(s)
18.1 Types	`<cstddef>`
18.2 Implementation properties	`<limits>`
18.4 Dynamic memory management	`<new>`
18.5 Type identification	`<typeinfo>`
18.6 Exception handling	`<exception>`