Accredited Standards Committee
X3, INFORMATION PROCESSING SYSTEMS[*]

Doc No: X3J16/92-0047
WG21/N0124
Date: May 22, 1992
Project: 738-D (PL C++)
Reply To: Stephen D. Clamage
steve@taumet.com

accept.

Wide Character as a Distinct Type

Stephen D. Clamage and Jerry Schwarz

Abstract

In Standard C, a compiler must treat the wide-character type as identical to one of the built-in integer types. When the programmer includes the standard header <stdlib.h>, the type becomes visible and given the name wchar_t. To make it possible to write portable C++ programs which overload functions based on wchar_t, we propose making wchar_t designate a unique type which must be implemented the same way as one of the standard integral types.

BACKGROUND:

In Standard C and in C++, a literal of the form L'x' has an implementation-dependent *wide-character* type which is identical to one of the standard integral types. A literal of the form L"string" has type *array-of-wide-character*. To find out the underlying type of a wide character, the programmer must include the standard header <stdlib.h>, which makes it available as a typedef with the name wchar_t. An implementation may choose any integral type for type wchar_t, and because it is a typedef, the actual type cannot be determined by a preprocessor test.

PROBLEM:

In C++ it would be desirable to be able to overload functions based on type wchar_t. Suppose we write

```
int foo(int);
int foo(wchar_t);
```

This code is legal if and only if type wchar_t is typedef'd to be something other than type int. There is no way to write a portable program having this overloading.

In the C++ iostream library it is essential to be able to distinguish wchar_t from all other types. For example, the actions of the following functions (among others) must be different:

```
ostream& operator<<(char);
ostream& operator<<(int);
ostream& operator<<(wchar_t);
```

[*] *Operating under the procedures of the American National Standards Institute (ANSI)*
Standards Secretariat: CBEMA, 1250 Eye St. NW, Suite 200, Washington, DC 20005

A character must be taken as-is. An integer must be converted to a string of digits in the current radix. A wide character must be converted to some implementation-dependent encoding suitable for output (probably a multi-byte string).

Under the current rules it is impossible to satisfy all of these requirements. It is necessary and sufficient to make wchar_t distinct from all other standard types.

SOLUTION:

We would like to avoid incompatibility with Standard C. We should require that wchar_t be implemented in the same way as one of the standard integral types, and specify corresponding type-promotion rules. This will allow a conforming Standard C program using wide characters to have the same semantics and work the same way when treated as a C++ program.

We propose the following:

1. Type wchar_t must have the same number of bits and the same signedness as one of the Standard C integral types. The C++ implementation must define which type this is. We will refer to this type as the *corresponding integral type.*

2. Type wchar_t is distinct from all other standard types. By way of example, type int usually has the same representation as either type short or long, yet it is always distinct from both.

3. Whenever type promotion is required (such as in an expression) type wchar_t is first implicitly promoted to its corresponding integral type. The other type-promotion rules apply thereafter.

ALTERNATIVES:

We cannot avoid making the wide-character type distinct from all other types, for reasons of overloading, as stated earlier.

We could make wchar_t a reserved word, but this would introduce an unnecessary incompatibility with C.

IMPACT ON IMPLEMENTATIONS:

Negligible. The compiler already must have an internal definition for the wide-character type. This proposal just means the compiler must maintain one additional pre-defined internal type for wchar_t. Standard library header <stdlib.h> may need some minor modifications.

IMPACT ON PROGRAMMERS:

There should be none.