
An alternative wide character type

X3J16/95-0084
WG21/N0684

Kevlin Henney
Westinghouse Systems Ltd
kevlin@wslint.demon.co.uk

1. INTRODUCTION

The current type for representing wide characters in C++ is 'wchar_t'. 'wchar_t' is a fundamental type constrained to have the implementation of one of the other integral types, but with its own literal form, eg.

```
        wchar_t a_wide_character = L'a';
        wchar_t a_wide_string[]  = L"a";
```

'wchar_t' is a key word, breaking with the tradition that the '_t' suffix indicates a 'typedef'. This is also against the 'spirit' of the language in which 'long' has previously been used to widen a fundamental type, eg. long int and long double.

The meaning of 'wchar_t' is not readily apparent to observers accustomed to programming with only 'char'. Obscuring the similarity between the two character precisions seems unnecessary and may be the source of some confusion for beginners.

This paper proposes that wide characters have the type 'long char' and 'wchar_t' is returned to its previous status as a reserved word. To achieve compatibility between C and C++ code the type of 'wchar_t' must be 'long char', eg.

```
        typedef long char wchar_t;
```

2. DISCUSSION

The requirement for representing character sets than cannot normally be held within a single 'char' was met quite late by the ANSI C standardization effort. A literal form was added, but the logical step of adding a new fundamental type was not taken. Instead 'wchar_t' was added as a new type name for an existing integral type.

At first sight this may seem analagous to the type names 'size_t' and 'ptrdiff_t'. However, neither of these types have separate literal forms. The notion of a new class of character constant seems to suggest something more primitive, along the lines of the 'long double' which was added as a fundamental type to C by ANSI.

This oversight had unfortunate repercussions in C++ when it was realised that functions taking 'wchar_t' could not be portably overloaded with functions accepting an arbitrary integral type. This is especially true for I/O where a wide character constant is expected from a 'wchar_t' and an numeric literal from other types of integer.

A class based solution is inadequate because of the literal form; only a fundamental type will do. The name 'wchar_t' was chosen for its obvious compatibility with C, but at the cost of a new keyword and some elegance.

The proposal to use 'long char' is more of a reduction than an extension to the language, simplifying it and restoring 'wchar_t' to a type synonym, but now with a mandated implementation. The constraint on the possible implementation of 'long char' remains:

```
        sizeof(long char) == sizeof(char) ||
        sizeof(long char) == sizeof(short int) ||
        sizeof(long char) == sizeof(int) ||
        sizeof(long char) == sizeof(long int)
```

The requirement, that each member of the C++ character set is represented by the same number in both character precisions, only strengthens the case that these two types should be seen to be more similar.

Another precedent for 'regularising' the language comes from the standardization of C in the form of the 'signed' keyword, allowing 'signed' and 'unsigned' forms of 'char' and 'int' bitfields to be differentiated portably.

However, it does not seem necessary to have separate 'signed' and 'unsigned' versions of 'long char', and this document does not propose these. Should these forms be required for completeness, the signedness could easily be patterned after 'char'.

In principle, withdrawing 'wchar_t' from the list of key words should not cause too many problems: 'wchar_t' would be defined as 'long char' in all the headers that used it, eg. <wstring>, and in the stand alone headers <stddef.h> and <cstddef>. The worst case change required for code built on the assumption that 'wchar_t' is to include <cstddef>; it seems unlikely that any transitional period of grace for compilers is required. The mandated implementation of 'wchar_t' as 'long char' ensures that no change to the existing definition of the standard libraries is required.

3. SUMMARY

Discussions, both face-to-face and over e-mail, seem to suggest that 'long char' is more in keeping with the spirit of the language. It seems an obvious improvement to make, and would rid the language of an unnecessary and unpleasant keyword.

4. PROPOSAL

The basic changes are listed here. If there is interest in this proposal then a list of precise changes to the most recent working paper will be added.

- The type for wide characters becomes ’long char’.

- ’wchar_t’ is withdrawn as a key word, but is still reserved.

- ’wchar_t’ is available as a synonym for ’long char’ in all headers in the standard library where it is currently referenced, and <stddef.h> and <cstddef>.

5. ACKNOWLEDGEMENTS

My thanks to Sean Corfield for his comments and for originally circulating this proposal on my behalf.