Document Number: WG14 N390/X3J11 94-075

C9X Revision Proposal
====================

Title: Extending character constants for named characters.
Author: Frank Farance
Author Affiliation: Farance Inc.
Postal Address: 555 Main Street, New York, NY, 10044-0150, USA
E-mail Address: frank@farance.com
Telephone Number: +1 212 486 4700
Fax Number: +1 212 759 1605
Sponsor: X3J11
Date: 1994-12-04
Proposal Category:
- __ Editorial change/non-normative contribution
- __ Correction
- X_ New feature
- __ Addition to obsolescent feature list
- __ Addition to Future Directions
- __ Other (please specify) ______________________________

Area of Standard Affected:
- X_ Environment
- X_ Language
- __ Preprocessor
- __ Library
    - __ Macro/typedef/tag name
    - __ Function
    - __ Header

Prior Art: BASIC language extended character constants.
Target Audience: Internationalization and terminal control.
Related Documents (if any): None.
Proposal Attached: __ Yes X_ No, but what's your interest?
Abstract:
Standard C has a large body of code that is based upon USA English. Even if a programmer wishes to write a program based upon some other language character set, e.g., Japanese, it is incovenient to do so because the programmer must acquire a compiler that understands the native character constants of the *run-time* environment. This extension allows the programmer to spell the names of characters using only ISO 646 characters, e.g., 'iso-8859-1-letter-a'. This feature is also useful for spelling terminal independent features, such as the UNIX TERMINFO sequences, e.g., 'terminfo-clear' (clears the screen). This extension does not propose to standardize these sequences, only to provide a feature for ``binding'' character set standards to C so that other standards efforts may describe how to ``spell'' they names of their characters. The programmer would include these character names by including the appropriate application header, e.g., "#include <iso10646.h>". The existing support in Standard C for wide character constants does not provide enough support because is requires the both the compiler and run-time system to understand the run-time character set -- this

extension only requires the run-time environment to understand the run-time character set.

The proposed extension creates a new type of character constant, ``multiple character constant'', which can be promoted to a string, a multibyte character string, a wide character constant, and a character constant.

```
        /* wide character constant */
        L'japanese-kana-ha'

        /* string paste gives multibyte character string */
        L""'japanese-kana-ha'

        /* character constant */
        'iso-8859-1-solidus'

        /* string */
        ""'terminfo-clear'
```

The multiple character constant extends the syntax of single character constants by (1) allowing multiple character names for characters constants:

```
        #define lsqbr 'iso-8859-1-left-bracket'
        #define rsqbr 'iso-8859-1-right-bracket'
        printf("array%c%d%c\n",lsqbr,17,rsqbr);
```

(2) allowing character constants to combine with string constants for string pasting:

```
        /* These are equivalent: */
        printf("array" lsqbr "%d" rsqbr "\n",17);
        printf("array[%d]\n",17);
```

(3) allowing multiple character constants (such as a ``clear screen'' sequence) to be promoted to string constants by pasting with the empty string "":

```
        /* Clears the screen. */
        printf("" 'terminfo-clear');
```

(4) promotes to an unsigned integer if used unadorned in an expression:

```
        char a,z;
        a = 'iso-8859-1-letter-a';
        z = 'iso-8859-1-letter-a'+25;
```

These features are especially useful for writing portable code for systems that interact with several character sets:

```
        unsigned int ascii_to_ebcdic[256] =
        {
                ['ascii-letter-a'] = 'ebcdic-letter-a',
                ['ascii-letter-b'] = 'ebcdic-letter-b',
                ['ascii-letter-c'] = 'ebcdic-letter-c',
                /* ... */
                ['ascii-letter-z'] = 'ebcdic-letter-z',
        };

        unsigned int ebcdic_to_ascii[256] =
        {
                ['ebcdic-letter-a'] = 'ascii-letter-a',
                ['ebcdic-letter-b'] = 'ascii-letter-b',
                ['ebcdic-letter-c'] = 'ascii-letter-c',
                /* ... */
                ['ebcdic-letter-z'] = 'ascii-letter-z',
        };
```

The proposal will be developed by investigating and specifying the following:

- Creating a new type of character constant.
- Interaction with existing multibyte character strings.
- Interaction with existing wide character constants.
- String pasting with character constants.
- Sample implementation to verify concepts.
- Interaction with existing library functions.