c73

(PAGE 1 OF 6)

ISO/TC97/SC22/WG14/N020

7) MULTIBYTE CHARACTERS

Here is my proposed wording for adding multi-byte character support:

CHANGE 1.6 DEFINITIONS OF TERMS (p. 3, l. 1) FROM:

o Byte - the unit of data storage in the execution environment large enogh to hold a single character in the character set of the execution environment.

TO:

o Byte - the unit of data storage in the execution environment large enogh to hold a single character in the (non-composite) character set of the execution environment.

ADD SAME SECTION (p. 3, l. 8):

o Multi-byte character - a sequence of one or more bytes representing a single character in the composite character set of either the source or the execution environment. Composite characters form a superset of the characters representable in a single byte.

CHANGE 2.2.1 Character sets (p. 11, l. 12) FROM:

A representation with all bits set to 0, called the *null character*, shall exist in the execution character set; it is used to terminate a string literal.

At least the following characters shall be in the source and execution character sets:

TO:

A single-byte representation with all bits set to 0, called the *null character*, shall exist in the execution character set; it is used to terminate a string literal.

At least the following single byte characters shall be in the source and execution character sets:

ADD 2.2.1.2 Multibyte characters (p. 12, l. 18):

The source character set may contain multi-byte characters, used to represent characters in a composite character set. The execution character set may also contain multi-byte characters, which need not have the same encoding as for the source character set. For each character set, the following shall hold:

- the single-byte characters defined in P2.2.1 shall be present.

- the presence, meaning, and representation of any additional characters is locale-specific.

- a multi-byte character may have a *state dependent encoding*, wherein each sequence of multi-byte characters begins in an *initial shift state* and enters other *shift states* when specific multi-byte-characters are encountered in the sequence. The interpretation of subsequent bytes in the sequence is a function of the current shift state.

- a byte with all bits zero shall be interpreted as a null character independent of shift state.

- a byte with all bits zero shall not occur in the second or subsequent bytes of a multi-byte character.

For the source character set, the following shall hold:

- a comment, string literal, or character constant shall begin in the initial shift state.

- a comment, string literal, or character constant shall consist of a sequence of complete multi-byte characters.

ADD TO 2.2.4.2 Numerical limits (p. 14, l. 20):

o maximum number of bytes in a multi-byte character, for any supported locale

MULTI_BYTE_MAX 1

CHANGE 3.1.3.4 Character constants (p. 26, l. 34) FROM:

A character constant is a sequence of one or more characters enclosed in single-quotes, as in 'x' or 'ab'.

TO:

A character constant is a sequence of one or more multi-byte characters enclosed in single-quotes, as in 'x' or 'ab'.

CHANGE 3.1.4 String literals (p. 28, l. 15) FROM:

A string literal is a sequence of zero or more characters enclosed in double-quotes, as in "xyz".

TO:

A string literal is a sequence of zero or more multi-byte characters enclosed in double-quotes, as in "xyz".

CHANGE 3.1.7 Comments (p. 29, l. 35) FROM:

The contents of a comment are examined only to find the characters */ that terminate it.[16]

TO:

The contents of a comment are examined only to identify multi-byte characters and to find the characters */ that terminate it.[16]

CHANGE 4.3 CHARACTER HANDLING <ctype.h> (p. 92, l. 2) FROM:

The header <ctype.h> declares several functions useful for testing and mapping characters.[74] In all cases the argument is an **int**, the value of which shall be representable as an **unsigned char** or shall be equal to the value of the macro **EOF**. If the argument has any other value, the behavior is undefined.

<STDLIB.H>
<stdlib.h>

TO:

The header ~~<ctype.h>~~ declares a type, a macro, and several functions useful for testing and mapping characters.[74]

The type is

    wchar_t

which is an integral type ~~whose size is at least as large as the size of **int**, and~~ whose range of values can represent distinct codes for all members of the largest composite character set specified among the supported locales. The null character shall have code value 0, each character defined in P2.2.1 shall have a code value equal to its value when used as the lone character in a character constant, and the value of the macro **EOF** shall be representable and distinct from the code for any member of any of the composite character sets specified among the supported locales.

The macro is

    multi_byte_max

which expands to an integer expression whose value is the maximum number of bytes in a multi-byte character for the composite character set specified by the current locale (category LC_CTYPE). Its value is at least 1 and never greater than the value of the macro MULTI_BYTE_MAX.

For the functions, in all cases the argument has type **wchar_t**, the value of which shall be a code for a character in the composite character set specified by the current locale, or shall equal the value of the macro **EOF**. If the argument has any other value, the behavior is undefined.

CHANGE SAME SECTION (various) FROM:

```
int isalnum(int c);
int isalpha(int c);
int iscntrl(int c);
int isdigit(int c);
int isgraph(int c);
int islower(int c);
int isprint(int c);
int ispunct(int c);
int isspace(int c);
int isupper(int c);
int isxdigit(int c);
int tolower(int c);
int toupper(int c);
```

TO:

```
int isalnum(wchar_t c);
int isalpha(wchar_t c);
int iscntrl(wchar_t c);
int isdigit(wchar_t c);
int isgraph(wchar_t c);
int islower(wchar_t c);
int isprint(wchar_t c);
int ispunct(wchar_t c);
int isspace(wchar_t c);
int isupper(wchar_t c);
int isxdigit(wchar_t c);
wchar_t tolower(wchar_t c);
wchar_t toupper(wchar_t c);
```

CHANGE 4.9.6.1 The fprintf function (p. 118, l. 40) FROM:

The format is composed of zero or more directives: ordinary characters (not %), which are copied unchanged to the output stream; ...

TO:

The format ~~is~~ shall be a multi-byte character sequence, beginning in its initial shift state. The format is composed of zero or more directives: ordinary multi-byte characters (not %), which are copied unchanged to the output stream; ...

CHANGE 4.9.6.2 The fscanf function (p. 121, l. 29) FROM:

The format ~~is~~ composed of zero or more directives: one or more white-space characters; an ordinary character (not %); or a conversion specification.

TO:

The format ~~is~~ shall be a multi-byte character sequence, beginning in its initial shift state. The format is composed of zero or more directives: one or more white-space characters; an ordinary multi-byte character (not %); or a conversion specification.

CHANGE SAME SECTION (p. 122, l. 5) FROM:

A directive that is an ordinary character is executed ...

TO:

A directive that is an ordinary multi-byte character is executed ...

ADD 4.10.7 Composite character functions (p. 146, l. 9):

~~[illegible]~~ multi-byte

4.10.7 Composite character functions

The behavior of the composite character functions is affected by changes in the LC_TYPE category of the locale. For a state dependent encoding, each function is placed in its initial state by a call with its argument **s** a null pointer. The function returns a non-zero value if encodings have state dependency, zero if there is no dependency. Subsequent calls with **s** other than a null pointer cause the internal state of the function to be altered as needed.

4.10.7.1 The mbtowc function

Synopsis

```
    #include <stdlib.h>
    int mbtowc(const char *s, size_t n, wchar_t *pwc);
```

Description

The **mbtowc** function determines the number of bytes comprising the multi-byte character pointed at by **s**. It then determines the code that corresponds to that multi-byte character for values of type **wchar_t**. If the multi-byte character is valid, and **pwc** is not a null pointer, then **mbtowc** stored the code in the object pointed to by **pwc**. At most **n** characters will be examined, starting at the character pointed to by **s**.

Returns

If **s** is a null pointer, then **mbtowc** returns a non-zero value if multi-byte characters have a state dependent encoding, or 0 if they do not. If the next **n** or fewer characters do not form a valid multi-byte character, **mbtowc** returns -1. If the next character is the null character, **mbtowc** returns 0. Otherwise, **mbtowc** returns the number of bytes that comprise the next multi-byte character. The value returned will then be greater than zero.

In no case will the value returned be greater than **n** or the value of the macro **multi_byte_max**.

4.10.7.1 The mblen function

Synopsis

```
    #include <stdlib.h>
    int mblen(const char *s, size_t n);
```

Description

The **~~mbtowc~~mblen** function determines the number of bytes comprising the multi-byte character pointed at by **s**. It behaves exactly like the function call:

```
    mbtowc(s, n, (wchar_t *)0);
```

4.10.7.3 The wctomb function

Synopsis

```
    #include <stdlib.h>
```

int wctomb(char *s, wchar_t wchar);

Description

The **wctomb** function determines the number of bytes needed to represent the multi-byte character corresponding to the code whose value is **wchar** (including any change in shift state). It stores the multi-byte character representation in the array object pointed to by **s**. At most **multi_byte_max** characters are stored.

Returns

If **s** is a null pointer, then **wctomb** returns a non-zero value if multi-byte characters have a state dependent encoding, or 0 if they do not. If the value of **wchar** is not a code corresponding to a valid multi-byte character, **wctomb** returns -1. If **wchar** has the value 0, **wctomb** returns 0. Otherwise, **wctomb** returns the number of bytes that comprise the multi-byte character corresponding to the code whose value is **wchar**. The value returned will then be greater than zero.

In no case will the value returned be greater than ~~**n** or the value of~~ the macro **multi_byte_max**.

CHANGE 4.12.3.5 The strftime function (p. 159, l. 13) FROM:

The format string consists of zero or more directives and ordinary characters. A directive consists of a % character followed by a character that determines the directive's behavior. All ordinary characters (including a terminating null character) are copied unchanged into the array.

TO:

The format is a multi-byte character sequence, beginning in its initial shift state. The format string consists of zero or more directives and ordinary multi-byte characters. A directive consists of a % character followed by a character that determines the directive's behavior. All ordinary multi-byte characters (including a terminating null character) are copied unchanged into the array.