NAME

mbrtowc – convert from multibyte to wide character encoding

SYNOPSIS

#include <wchar.h>

size_t mbrtowc(wchar_t * *pwc*, **const char *** *s*, **size_t** *n*, **mbstate_t *** *ps* **);**

Feature Test Macro Requirements for libmingwex:

__MSVCRT_VERSION__: since mingwrt-5.3, if this feature test macro is *defined*, with a value of *at least 0x0800*, (corresponding to the symbolic constant, **__MSCVR80_DLL**, and thus declaring intent to link with MSVCR80.DLL, or any later version of Microsoft's non-free runtime library, instead of with MSVCRT.DLL), calls to **mbrtowc**() will be directed to the implementation thereof, within Microsoft's runtime DLL.

_ISOC99_SOURCE, **_ISOC11_SOURCE**: since mingwrt-5.3.1, when linking with MSVCRT.DLL, or when **__MSVCRT_VERSION__** is either *undefined*, or is *defined* with any value which is *less than 0x0800*, (thus denying intent to link with MSVCR80.DLL, or any later non-free version of Microsoft's runtime library), *explicitly* defining either of these feature test macros will cause any call to **mbrtowc**() to be directed to the *libmingwex* implementation; if neither macro is defined, calls to **mbrtowc**() will be directed to Microsoft's runtime implementation, if it is available, otherwise falling back to the libmingwex implementation.

Prior to mingwrt-5.3, none of the above feature test macros have any effect on **mbrtowc**(); all calls will be directed to the libmingwex implementation.

DESCRIPTION

If *s* is a NULL pointer, the **pwc*, and the *n* arguments are ignored, and the call to **mbrtowc**() function is interpreted as if invoked as

```
mbrtowc( NULL, "", 1, ps );
```

Otherwise, if *s* is not a NULL pointer, the **mbrtowc**() function inspects the sequence of bytes, starting at *s*, up to a maximum of *n* bytes, to determine the number of bytes required to complete the next multibyte code point, commencing from the conversion state specified in **ps*, (which is then updated). Then, if **pwc* is not a NULL pointer, and *n* or fewer bytes is sufficient to complete a single multibyte character, the single **wchar_t** wide character conversion of that multibyte character is stored at **pwc*.

The sequence of bytes, pointed to by *s*, is interpreted as a multibyte character sequence in the codeset which is associated with the **LC_CTYPE** category of the active process locale.

If *ps* is specified as a NULL pointer, **mbrtowc**() will track conversion state using an internal **mbstate_t** object reference, which is private within the **mbrtowc**() process address space; at process start-up, this internal **mbstate_t** object is initialized to represent the initial conversion state.

In the special case, where the conversion of a completed multibyte character must be represented as a **UTF-16LE** *surrogate pair*, and **pwc* is not a NULL pointer, only the *high surrogate* will be stored at **pwc*; please refer to the section **CAVEATS AND BUGS**, below, for advice on retrieval of the *low surrogate*.

RETURN VALUE

If the multibyte sequence, completed by *n* or fewer bytes, does not represent the NUL code point, then **mbrtowc**() returns the number of bytes which are actually required to complete the sequence, (a number between 1 and *n*, inclusive), and the conversion state, as specified in **ps*, is reset to the initial state; if *pwc* is not a NULL pointer, the wide character conversion of the completed multibyte character is stored at **pwc*.

On the other hand, if the completed multibyte sequence *does* represent the NUL code point, then **mbrtowc**() returns zero, and the conversion state, as specified in **ps*, is reset to the initial state; if *pwc* is not a NULL pointer, the NUL wide character is stored at **pwc*.

If *n* is less than the effective **MB_CUR_MAX** for the active process locale, and *n* bytes is insufficient to complete a multibyte character, then **ps* is updated to represent a new partially completed encoding state, (no wide character conversion is stored), and **mbrtowc**() returns *(size_t)(−2)*. (If *n* is equal to, or greater than **MB_CUR_MAX**, this return condition can arise, only if the multibyte encoding sequence includes redundant shift states; since shift states are not used, this cannot occur in any MS-Windows multibyte character set).

ERROR CONDITIONS

If the sequence of *n* or fewer bytes, pointed to by *s*, extends any pending encoding state recorded within **ps*, to at least **MB_CUR_MAX** bytes, and the resulting sequence does not represent a valid multibyte character, then *errno* is set to **EILSEQ**, no wide character conversion is stored, and **mbrtowc**() returns *(size_t)(−1)*.

Conforming to POSIX.1, as an extension to ISO-C99, if, on entry to **mbrtowc**(), the conversion state represented by **ps* is deemed to be *invalid*, *errno* is set to **EINVAL**, and **mbrtowc**() returns *(size_t)(−1)*; the conversion state may be deemed to be invalid if it contains any sequence of bytes which does not match a valid initial sequence from a multibyte character representation within the currently active codeset, if it can be interpreted as a complete multibyte character, *without* the addition of any further bytes from *s*, or if it represents a *surrogate pair* conversion, resulting from a preceding call to **mbrtowc**(), from which the *low surrogate* has yet to be retrieved, (and this is not the special case in which *n* is specified as *zero*, indicating that this call is intended to retrieve that pending *low surrogate*).

CONFORMING TO

Except in respect of its extended provision for handling of *surrogate pairs*, and to the extent that it may be affected by limitations of the underlying MS-Windows API, the libmingwex implementation of **mbrtowc**() conforms generally to ISO-C99, POSIX.1-2001, and POSIX.1-2008; (prior to mingwrt−5.3, and in those cases where calls may be delegated to a Microsoft runtime DLL implementation, this level of conformity may not be achieved).

CAVEATS AND BUGS

Due to a documented limitation of Microsoft's **setlocale**() function implementation, it is not possible to directly select an active locale, in which the codeset is represented by any multibyte character sequence with an effective **MB_CUR_MAX** of more than two bytes. Prior to *mingwrt-5.3*, this limitation precludes the use of **mbrtowc**() to interpret any codeset with **MB_CUR_MAX** greater than two bytes, (such as **UTF-8**). From *mingwrt-5.3* onward, the MinGW.org implementation of **mbrtowc**() mitigates this limitation by assignment of the codeset from the **LC_CTYPE** environment variable, provided the system default has been previously activated for the **LC_CTYPE** locale category; e.g. execution of:

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <limits.h>
#include <wchar.h>

void print_conv( const char *mbs )
{
  wchar_t wch;
  size_t n = mbrtowc( &wch, mbs, MB_LEN_MAX, NULL );
  if( (int)(n) > 0 ) printf( "%u bytes -> 0x%04X\n", n, wch );
  else if( n == (size_t)(-1) ) perror( "mbrtowc" );
}

int main()
{
  setlocale( LC_CTYPE, "" );
  putenv( "LC_CTYPE=en_GB.65001" );
  print_conv( "\U0001d10b" );
```

```
  print_conv( "\u6c34" );
  return 0;
}
```

will interpret the string "\U0001d10b" as a four-byte **UTF-8** encoding sequence, (which represents a single Unicode code point), but will fail to interpret the following "\u6c34" sequence, (which also represents a valid Unicode code point), and, (if **stderr** is redirected to **stdout**), will print the result as:

```
4 bytes -> 0xD834
mbrtowc: Invalid argument
```

This example illustrates a potentially irreconcilable deviation of any **mbrtowc**() implementation, on MS-Windows, from the ISO-C99 standard: due to Microsoft's choice of **UTF-16LE** as the underlying representation of the **wchar_t** data type, it is not possible to satisfy the requirement, implicit in the ISO-C99 specification for **mbrtowc**(), that it should be possible to return the complete representation of any single representable Unicode code point as a single **wchar_t** value. In the case of this example, whereas the 4-byte **UTF-8** representation of the "\U0001d10b" Unicode code point *is* complete, the 0xD834 **wchar_t** representation, as returned by **mbrtowc**(), is *not* complete; it represents a **UTF-16** *high surrogate*, which *must* be paired with a corresponding *low surrogate* to complete it, and, since ISO-C99 requires that the ***pwc** argument to **mbrtowc**() refers to sufficient storage space to accommodate only *one* **wchar_t** value, it is not possible for **mbrtowc**() to *safely* return *both* the *high surrogate*, and its complementary *low surrogate*, in a single call. To mitigate this non-conformance, from mingwrt-5.3 onward, the MinGW implementation of **mbrtowc**() supports the following non-standard strategy for completion of any conversion which requires return of a *surrogate pair*:

- Any translation unit, in which **mbrtowc**() is called, should:
  a) explicitly define either the **_ISOC99_SOURCE**, or the **_ISOC11_SOURCE** feature test macro, (with any arbitrary value, or even no value), **before** including *any* header file, and
  b) include the **<winnls.h>** header file, in addition to the required **<wchar.h>** header.
- Following each call of **mbrtowc**(), which returns a **wchar_t** value with a converted byte count greater than zero, test the returned **wchar_t** value, using the **IS_HIGH_SURROGATE**() macro.
- When the **IS_HIGH_SURROGATE**() macro call indicates that the returned **wchar_t** value does represent a *high surrogate*, immediately call **mbrtowc**() again, passing the ***ps** state as returned by the original call, together with the original multibyte sequence reference, but with an explicit scan length limit, **n**, of zero, and an alternative **wchar_t** buffer reference pointer, for storage of the *low surrogate*; on successful retrieval of this *low surrogate*, the additional converted byte count will be returned as zero, and the pending ***ps** conversion state will have been cleared, (i.e. reset to the initial state).

Thus, considering the preceding example, to support interpretation of *surrogate pairs* the example code should be modified by insertion of:

```
#define _ISOC99_SOURCE
#include <winnls.h>
```

at the top of the source file, and reimplementation of the **print_conv**() function, to incorporate the **IS_HIGH_SURROGATE**() test, and response:

```
void print_conv( const char *mbs )
{
  wchar_t wch;
  size_t n = mbrtowc( &wch, mbs, MB_LEN_MAX, NULL );
  if( (int)(n) > 0 )
  {
    if( IS_HIGH_SURROGATE( wch )
    {
      wchar_t wcl;
      mbrtowc( &wcl, mbs, 0, NULL );
```

```
        printf( "%u bytes -> 0x%04X:0x%04X\n", n, wch, wcl );
      }
      else printf( "%u bytes -> 0x%04X\n", n, wch );
    }
    else if( n == (size_t)(-1) ) perror( "mbrtowc" );
  }
```

With these changes in place, the output from the program becomes:

```
4 bytes -> 0xD834:0xDD0B
3 bytes -> 0x6C34
```

thus now correctly reporting the conversion of the *surrogate pair*, and then correctly interpreting the following 3-byte **UTF-8** sequence.

Please be aware that the underlying MS-Windows API, which is used to interpret the multibyte sequence, offers no readily accessible mechanism to discriminate between incomplete and invalid sequences; thus, if *n* is less than the effective **MB_CUR_MAX** for the active codeset, this **mbrtowc**() implementation may return *(size_t)(−2)*, indicating an incomplete sequence, even in cases where there are no additional bytes which could be appended, to complete a valid encoding sequence.

SEE ALSO

mbsrtowcs(3)

AUTHOR

This manpage was written by Keith Marshall, <keith@users.osdn.me>, to document the **mbrtowc**() function as it has been implemented for the MinGW.org Project.