IBM Education Assistance for z/OS V2R1

Item: N1040 Functions
Element/Component: Language Environment

Material is current as of June 2013

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Presentation Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Presentation Objectives

- Explain the purpose of the new functions:
  - mbrtoc16(), mbrtoc32()
  - c16rtomb(), c32rtomb()
- Explain how the new functions are used

Overview

- The latest C programming language standard, ISO/IEC 9899:2011(C11), adds in two new types char16_t, char32_t together with new syntax for character and string literals of those types, and four new functions mbrtoc16(), mbrtoc32(), c16rtomb(), c32rtomb(). They enhance the C language for manipulating the Unicode characters.
- Problem Statement / Need Addressed
  - Currently the XL C/C++ runtime library and compiler already define and support the two types, but the library does not provide the APIs for applications to perform related character conversions.
  - International applications usually have data in Unicode encoding, while traditional APIs of the C/C++ runtime library are designed to handle multibyte characters defined by locales. The four new functions help to close the gap between Unicode and multibyte characters.

Overview

- Solution
  - Implement four new functions, mbrtoc16(), mbrtoc32(), c16rtomb() and c32rtomb() as specified in C11.
- Benefit / Value
  - Users can use these four functions when they want to convert between multibyte characters and specified Unicode characters.

Usage & Invocation (mbrtoc16() / mbrtoc32())

- Convert a multibyte character to a char16_t / char32_t character.
- ```
  #include <uchar.h>
  size_t mbrtoc16(char16_t * restrict pc16,
                  const char * restrict s,
                  size_t n,
                  mbstate_t * restrict ps);
  size_t mbrtoc32(char32_t * restrict pc32,
                  const char * restrict s,
                  size_t n,
                  mbstate_t * restrict ps);
  ```
- pc16: address of output char16_t character buffer.
  pc32: address of output char32_t character buffer.
- s: address of input multibyte characters.
- n: number of bytes that mbrtoc16() can inspect at most.
- ps: conversion state object.
- mbrtoc32() has similar usage to mbrtoc16(). The following pages explain mbrtoc16() only.
  ```

Usage & Invocation (mbrtoc16() / mbrtoc32())

```
#include <uchar.h>
size_t mbrtoc16(char16_t * restrict pc16,
                const char * restrict s,
                size_t n,
                mbstate_t * restrict ps);
```

- If **s** is not a null pointer, the mbrtoc16() function inspects at most **n** bytes beginning with the byte pointed to by **s** to determine the number of bytes needed to complete the next multibyte character (including any shift sequences).
- The mbrtoc16() function returns the first of the following that applies (given the current conversion state):
  - 0: if the next **n** or fewer bytes complete the multibyte character that corresponds to the null wide character (which is the value stored).
  - Between 1 and n inclusive: if the next **n** or fewer bytes complete a valid multibyte character (which is the value stored); the value returned is the number of bytes that complete the multibyte character.

Usage & Invocation (mbrtoc16() / mbrtoc32())

```
#include <uchar.h>
size_t mbrtoc16(char16_t * restrict pc16,
                const char * restrict s,
                size_t n,
                mbstate_t * restrict ps);
```

- Continue
  - -3 : if the next character resulting from a previous call has been stored (no bytes from the input have been consumed by this call).
  - -2 : if the next **n** bytes contribute to an incomplete (but potentially valid) multibyte character, and all **n** bytes have been processed (no value is stored).
  - -1 : if an encoding error occurs, in which case the next **n** or fewer bytes do not contribute to a complete and valid multibyte character (no value is stored); the value of the macro EILSEQ is stored in errno, and the conversion state is unspecified.

Usage & Invocation (mbrtoc16() / mbrtoc32())

- Example

```
#include <uchar.h>
int main(void)
{
   char16_t c16;
   char mbs[]   = "a" ; /* string containing the multibyte char  */
   mbstate_t ss = 0 ;   /* set shift state to the initial state */
   int length   = 0 ;
   length = mbrtoc16(&c16, mbs, MB_CUR_MAX, &ss);
   if (length < 0) {
      perror("mbrtoc16() fails to convert");
      exit(-1);
   }
   printf(" mbs:\"%s\"\n", mbs);
   printf(" length: %d \n", length);
   printf(" c16: 0x%04x \n", c16);
}
```

- Output:

```
mbs:"a"
length: 1
c16: 0x0061
```

Usage & Invocation (mbrtoc16() / mbrtoc32())

- ```
  #include <uchar.h>
  size_t mbrtoc16(char16_t * restrict pc16,
                  const char * restrict s,
                  size_t n,
                  mbstate_t * restrict ps);
  ```
- Usage notes:
  - If s is a null pointer, the mbrtoc16() function is equivalent to the call:
    ```
    mbrtoc16(NULL, "", 1, ps)
    ```
    In this case, the values of the parameters pc16 and n are ignored.
  - If ps is a null pointer, mbrtoc16() uses its own internal object to track the shift state. Otherwise *ps must be a valid mbstate_t object.
  - To use this function, you must compile with LANGLVL(EXTC1X).
  - mbrtoc16() only supports the codesets(CCSIDs) provided by Unicode Service.
  - The result of converting multiple string alternately in one thread by using multiple mbstate_t objects (including the internal one) is undefined.
    ```

Usage & Invocation (c16rtomb() / c32rtomb())

- Convert a char16_t / char32_t character to a multibyte character.
- ```
  #include <uchar.h>
  size_t c16rtomb(char  * restrict s,
                  char16_t c16,
                  mbstate_t * restrict ps);
  size_t c32rtomb(char  * restrict s,
                  char32_t c32,
                  mbstate_t * restrict ps);
  ```
- s : address of output multibyte character buffer
- c16: input char16_t character
  c32: input char32_t character
- ps: conversion state object.
- c32rtomb() has similar usage to c16rtomb(). The following pages explain c16rtomb() only.
  ```

Usage & Invocation (c16rtomb() / c32rtomb())

- ```
  #include <uchar.h>
  size_t c16rtomb(char  * restrict s,
                  char16_t c16,
                  mbstate_t * restrict ps);
  ```
- If s is not a null pointer, the c16rtomb() function determines the number of bytes needed to represent the multibyte character that corresponds to the wide character given by c16 (including any shift sequences), and stores the multibyte character representation in the array whose first element is pointed to by s. At most MB_CUR_MAX bytes are stored.
- The c16rtomb() function returns the number of bytes stored in the array object (including any shift sequences).
- When c16 is not a valid wide character, an encoding error occurs: the function stores the value of the macro EILSEQ in errno and returns (size_t)(-1); the conversion state is unspecified.
  ```

Usage & Invocation (c16rtomb() / c32rtomb())

- Example

```
#include <uchar.h>
int main(void)
{
   char16_t in  = u'a';
   mbstate_t st = 0;
   char out[MB_CUR_MAX];
   int rc, i;
   rc = c16rtomb(out, in, &st);
   if (rc < 0) {
      perror("c16rtomb() fails to convert");
      exit(-1);
   }
   printf(" c16: 0x%04x \n",   in);
   printf(" return code: %d \n", rc);
   printf(" mb character: ");
   for (i=0; i<rc; i++)
      printf(" 0x%02x", out[i]);
   printf("\n");
   return 0;
}
```

- Output

```
c16: 0x0061
return code: 1
mb character: 0x81
```

Usage & Invocation (c16rtomb() / c32rtomb())

- ```
  #include <uchar.h>
  size_t c16rtomb(char  * restrict s,
                  char16_t c16,
                  mbstate_t * restrict ps);
  ```
- Usage notes:
  - If s is a null pointer, the c16rtomb() function is equivalent to the call:
    ```
    c16rtomb(buf, L'\0', ps)
    ```
    where buf is an internal buffer.
  - If ps is a null pointer, c16rtomb() uses its own internal object to track the shift state. Otherwise *ps must be a valid mbstate_t object.
  - To use this function, compile the source code with LANGLVL(EXTC1X).
  - c16rtomb() only supports the codesets(CCSIDs) provided by Unicode Service.
  - The result of converting multiple string alternately in one thread by using multiple mbstate_t objects (including the internal one) is undefined.
    ```

Usage & Invocation (c16rtomb() / c32rtomb())

- ```
  #include <uchar.h>
  size_t c16rtomb(char  * restrict s,
                  char16_t c16,
                  mbstate_t * restrict ps);
  ```
- More Usage notes:
  - The result s for stateful multibyte encodings, such as EBCDIC MBCS, may leave out shift bytes according to the conversion state. The first DBCS character in the output sequence has only shift-out character, and the following characters have neither shift-out nor shift-in. The ending shift-in will not be produced until a SBCS character or a null wide character is encountered.
  - The Unicode combining characters are not supported, and will be converted to substitute character of target CCSID.
  ```

Usage & Invocation (mbstate_t)

- mbstate_t is the type of conversion state object, that can completely describe the current conversion state of the associated multibyte character sequence, which the functions alter as necessary.
- Two ways to initialize a mbstate_t object (we take mbrtoc16() as an example).
  - Assign 0 to it.

```
mbstate_t state;
state = 0;
```

  - Call mbrtoc16() with NULL as output buffer.

```
mbstate_t state;
mbrtoc16(NULL, "", 1, &state);
```

  - Note that the internal object can only be initialized with the second method.

```
mbrtoc16(NULL, "", 1, NULL);
```

Presentation Summary

- The mbrtoc16(), mbrtoc32(), c16rtomb() and c32rtomb() functions are now available in XL C/C++ runtime library.

Appendix

- Publication References
  - z/OS XL C/C++ Programming Guide (SC09-4765)
  - z/OS XL C/C++ Runtime Library Reference (SA22-7821)