出自：<https://www.cnblogs.com/ranjiewen/p/5770639.html>

# [MultiByteToWideChar和WideCharToMultiByte用法详解](https://www.cnblogs.com/ranjiewen/p/5770639.html)

**注意：**

这两个函数是由Windows提供的转换函数，不具有通用性

C语言提供的转换函数为mbstowcs()/wcstombs()

**一、函数简单介绍**

涉及到的头文件：

函数所在头文件：windows.h

#include <windows.h>

wchar\_t类型所需头文件：wchar.h

#include <wchar.h>

( 1 ) MultiByteToWideChar()

函数功能：该函数映射一个字符串到一个宽字符（unicode）的字符串。由该函数映射的字符串没必要是多字节字符组。

函数原型：

int MultiByteToWideChar(

　　UINT CodePage,

　　DWORD dwFlags,

　　LPCSTR lpMultiByteStr,

　　int cchMultiByte,

　　LPWSTR lpWideCharStr,

　　int cchWideChar

　　);

参数：

1> CodePage：指定执行转换的多字节字符所使用的字符集

这个参数可以为系统已安装或有效的任何字符集所给定的值。你也可以指定其为下面的任意一值：

|  |  |
| --- | --- |
| **Value** | **Description** |
| CP\_ACP | ANSI code page |
| CP\_MACCP | Not supported |
| CP\_OEMCP | OEM code page |
| CP\_SYMBOL | Not supported |
| CP\_THREAD\_ACP | Not supported |
| CP\_UTF7 | UTF-7 code page |
| CP\_UTF8 | UTF-8 code page |

2> dwFlags：一组位标记，用以指出是否未转换成预作或宽字符（若组合形式存在），是否使用象形文字替代控制字符，以及如何处理无效字符。你可以指定下面是标记常量的组合，含义如下：

　　MB\_PRECOMPOSED：通常使用预作字符——就是说，由一个基本字符和一个非空字符组成的字符只有一个单一的字符值。这是缺省的转换选择。不能与MB\_COMPOSITE值一起使用。

　　MB\_COMPOSITE：通常使用组合字符——就是说，由一个基本字符和一个非空字符组成的字符分别有不同的字符值。不能与MB\_PRECOMPOSED值一起使用。

　　MB\_ERR\_INVALID\_CHARS：如果函数遇到无效的输入字符，它将运行失败，且GetLastErro返回ERROR\_NO\_UNICODE\_TRANSLATION值。

　　MB\_USEGLYPHCHARS：使用象形文字替代控制字符。

组合字符由一个基础字符和一个非空字符构成，每一个都有不同的字符值。每个预作字符都有单一的字符值给基础/非空字符的组成。在字符è中，e就是基础字符，而重音符标记就是非空字符。

标记MB\_PRECOMPOSED和MB\_COMPOSITE是互斥的，而标记MB\_USEGLYPHCHARS和MB\_ERR\_INVALID\_CHARS则不管其它标记如何都可以设置。

一般不使用这些标志，故取值为0时。

3> lpMultiByteStr：指向待转换的字符串的缓冲区。

4> cchMultiByte：指定由参数lpMultiByteStr指向的字符串中字节的个数。可以设置为-1，会自动判断lpMultiByteStr指定的字符串的长度

（如果字符串不是以空字符中止，设置为-1可能失败，可能成功），此参数设置为0函数将失败。

5> lpWideCharStr：指向接收被转换字符串的缓冲区。

6> cchWideChar：指定由参数lpWideCharStr指向的缓冲区的宽字节数。若此值为0，函数不会执行转换，而是返回目标缓存lpWideChatStr所需的宽字符数。

返回值：

如果函数运行成功，并且cchWideChar不为0，返回值是由lpWideCharStr指向的缓冲区中写入的宽字符数；

如果函数运行成功，并且cchMultiByte为0，返回值是待转换字符串的缓冲区所需求的宽字符数大小。（此种情况用来获取转换所需的wchar\_t的个数）

如果函数运行失败，返回值为零。

若想获得更多错误信息，请调用GetLastError()函数。它可以返回下面所列错误代码：

　　ERROR\_INSUFFICIENT\_BUFFER；     ERROR\_INVALID\_FLAGS；

　　ERROR\_INVALID\_PARAMETER；         ERROR\_NO\_UNICODE\_TRANSLATION。

( 2 ) WideCharToMultiByte()

函数功能：该函数映射一个unicode字符串到一个多字节字符串。

函数原型：

int WideCharToMultiByte(

　　UINT **CodePage**,

　　DWORD **dwFlags**,

　　LPCWSTR **lpWideCharStr**,

　　int **cchWideChar**,

　　LPSTR **lpMultiByteStr**,

　　int **cchMultiByte**,

　　LPCSTR **lpDefaultChar**,

　　LPBOOL **pfUsedDefaultChar**

　　);

参数：

与MultiByteToWideChar()函数中的参数类似，但是多了两个参数：

**lpDefaultChar**和**pfUsedDefaultChar**：只有当WideCharToMultiByte函数遇到一个宽字节字符，而该字符在uCodePage参数标识的代码页中并没有它的表示法时，WideCharToMultiByte函数才使用这两个参数。（通常都取值为NULL）

1> 如果宽字节字符不能被转换，该函数便使用lpDefaultChar参数指向的字符。如果该参数是NULL（这是大多数情况下的参数值），那么该函数使用系统的默认字符。该默认字符通常是个问号。这对于文件名来说是危险的，因为问号是个通配符。

2> pfUsedDefaultChar参数指向一个[布尔变量](http://baike.baidu.com/view/2792492.htm" \t "_blank)，如果Unicode字符串中至少有一个字符不能转换成等价多字节字符，那么函数就将该变量置为TRUE。如果所有字符均被成功地转换，那么该函数就将该变量置为FALSE。当函数返回以便检查宽字节字符串是否被成功地转换后，可以测试该变量。

**返回值**：

如果函数运行成功，并且cchMultiByte不为零，返回值是由 lpMultiByteStr指向的缓冲区中写入的字节数；

如果函数运行成功，并且cchMultiByte为零，返回值是接收到待转换字符串的缓冲区所必需的字节数。（此种情况用来获取转换所需Char的个数）

如果函数运行失败，返回值为零。

若想获得更多错误信息，请调用GetLastError函数。它可以返回下面所列错误代码：

　　ERROR\_INSUFFICIENT\_BJFFER；ERROR\_INVALID\_FLAGS；

　　ERROR\_INVALID\_PARAMETER；ERROR\_NO\_UNICODE\_TRANSLATION。

**二、使用方法**

( 1 ) 将多字节字符串转为宽字符串：

1) 调用MultiByteToWideChar()函数，设置cchWideChar参数为0（用以获取转换所需的接收缓冲区大小）；

2) 获取输入缓存的大小，作为cchMultiByte的值；（这样做是为了节省空间，也可以给cchMultiByte取值-1（字符串需要以空字符结尾，否则会出错））

3) 分配足够的内存块，用于存放转换后的Unicode字符串；

该内存块的大小由前面对cchWideChar()函数的返回值来决定；（也可以用别的方法，但该方法更节省内存）

4) 再次调用MultiByteToWideChar()函数，这次将缓存的地址作为lpWideCharStr,参数来传递，并传递第一次调用MultiByteToWideChar()函数时的返回值作为cchWideChar参数的值；

5) 使用转换后的字符串；

6) 释放接收缓冲区占用的内存块；

示例代码：

[复制代码](javascript:void(0);)

void main()

{

char sBuf[25]={0};

strcpy(sBuf, "我最棒");

//获取输入缓存大小

int sBufSize=strlen(sBuf);

//获取输出缓存大小

//VC++ 默认使用ANSI，故取第一个参数为CP\_ACP

DWORD dBufSize=MultiByteToWideChar(CP\_ACP, 0, sBuf, sBufSize, NULL, 0);

printf("需要wchar\_t%u个\n", dBufSize);

wchar\_t \* dBuf=new wchar\_t[dBufSize];

wmemset(dBuf, 0, dBufSize);

//进行转换

int nRet=MultiByteToWideChar(CP\_ACP, 0, sBuf, sBufSize, dBuf, dBufSize);

if(nRet<=0)

{

cout<<"转换失败"<<endl;

DWORD dwErr=GetLastError();

switch(dwErr)

{

case ERROR\_INSUFFICIENT\_BUFFER:

printf("ERROR\_INSUFFICIENT\_BUFFER\n");

break;

case ERROR\_INVALID\_FLAGS:

printf("ERROR\_INVALID\_FLAGS\n");

break;

case ERROR\_INVALID\_PARAMETER:

printf("ERROR\_INVALID\_PARAMETER\n");

break;

case ERROR\_NO\_UNICODE\_TRANSLATION:

printf("ERROR\_NO\_UNICODE\_TRANSLATION\n");

break;

}

}

else

{

cout<<"转换成功"<<endl;

cout<<dBuf;

}

delete(dBuf);

}

[复制代码](javascript:void(0);)

注意：两次调用MultiCharToWideChar()时，形参cchMultiByte的取值需要相同，否则可能会出现接收缓存不足之类的错误，从而导致转换失败！

 ( 2 ) 从宽字节转为窄字节字符串

步骤与（1）类似，故不赘述

代码示例如下：

[复制代码](javascript:void(0);)

//从宽字符串转换窄字符串

wchar\_t sBuf[25]={0};

wcscpy(sBuf, L"我最棒");

//获取转换所需的目标缓存大小

DWORD dBufSize=WideCharToMultiByte(CP\_OEMCP, 0, sBuf, -1, NULL,0,NULL, FALSE);

//分配目标缓存

char \*dBuf = new char[dBufSize];

memset(dBuf, 0, dBufSize);

//转换

int nRet=WideCharToMultiByte(CP\_OEMCP, 0, sBuf, -1, dBuf, dBufSize, NULL, FALSE);

if(nRet<=0)

{

printf("转换失败\n");

}

else

{

printf("转换成功\nAfter Convert: %s\n", dBuf);

}

delete []dBuf;

[复制代码](javascript:void(0);)

**三、MultiByteToWideChar()函数乱码的问题**

有的朋友可能已经发现，在标准的WinCE4.2或WinCE5.0 SDK模拟器下，这个函数都无法正常工作，其转换之后的字符全是乱码！

及时更改MultiByteToWideChar()参数也依然如此。不过这个不是代码问题，其结症在于所定制的操作系统.如果我们定制的操作系统默认语言不是中文，也会出现这种情况。

由于标准的SDK默认语言为英文，所以肯定会出现这个问题。而这个问题的解决，不能在简单地更改控制面板的"区域选项"的"默认语言"，而是要在系统定制的时候，选择默认语言为"中文"。系统定制时选择默认语言的位置于:   Platform -> Setting... -> locale -> default language ,选择"中文",然后编译即可。

Unicode ：宽字节字符集   
1. 如何取得一个既包含单字节字符又包含双字节字符的字符串的字符个数？   
可以调用Microsoft Visual C++的运行期库包含函数\_mbslen来操作多字节（既包括单字节也包括双字节）字符串。   
调用strlen函数，无法真正了解字符串中究竟有多少字符，它只能告诉你到达结尾的0之前有多少个字节。   
2. 如何对DBCS（双字节字符集）字符串进行操作？   
函数 描述   
PTSTR CharNext （ LPCTSTR ）; 返回字符串中下一个字符的地址   
PTSTR CharPrev （ LPCTSTR, LPCTSTR ）； 返回字符串中上一个字符的地址   
BOOL IsDBCSLeadByte( BYTE )； 如果该字节是DBCS字符的第一个字节，则返回非0值   
3. 为什么要使用Unicode？   
（1） 可以很容易地在不同语言之间进行数据交换。   
（2） 使你能够分配支持所有语言的单个二进制.exe文件或DLL文件。   
（3） 提高应用程序的运行效率。   
Windows 2000是使用Unicode从头进行开发的，如果调用任何一个Windows函数并给它传递一个ANSI字符串，那么系统首先要将字符串转换成Unicode，然后将Unicode字符串传递给操作系统。如果希望函数返回ANSI字符串，系统就会首先将Unicode字符串转换成ANSI字符串，然后将结果返回给你的应用程序。进行这些字符串的转换需要占用系统的时间和内存。通过从头开始用Unicode来开发应用程序，就能够使你的应用程序更加有效地运行。   
Windows CE 本身就是使用Unicode的一种操作系统，完全不支持ANSI Windows函数   
Windows 98 只支持ANSI，只能为ANSI开发应用程序。   
Microsoft公司将COM从16位Windows转换成Win32时，公司决定需要字符串的所有COM接口方法都只能接受Unicode字符串。   
4. 如何编写Unicode源代码？   
Microsoft公司为Unicode设计了WindowsAPI，这样，可以尽量减少代码的影响。实际上，可以编写单个源代码文件，以便使用或者不使用Unicode来对它进行编译。只需要定义两个宏（UNICODE和\_UNICODE），就可以修改然后重新编译该源文件。   
\_UNICODE宏用于C运行期头文件，而UNICODE宏则用于Windows头文件。当编译源代码模块时，通常必须同时定义这两个宏。   
5. Windows定义的Unicode数据类型有哪些？   
数据类型 说明   
WCHAR Unicode字符   
PWSTR 指向Unicode字符串的指针   
PCWSTR 指向一个恒定的Unicode字符串的指针   
对应的ANSI数据类型为CHAR，LPSTR和LPCSTR。   
ANSI/Unicode通用数据类型为TCHAR，PTSTR,LPCTSTR。   
6. 如何对Unicode进行操作？   
字符集 特性 实例   
ANSI 操作函数以str开头 strcpy   
Unicode 操作函数以wcs开头 wcscpy   
MBCS 操作函数以\_mbs开头 \_mbscpy   
ANSI/Unicode 操作函数以\_tcs开头 \_tcscpy（C运行期库）   
ANSI/Unicode 操作函数以lstr开头 lstrcpy（Windows函数）   
所有新的和未过时的函数在Windows2000中都同时拥有ANSI和Unicode两个版本。ANSI版本函数结尾以A表示；Unicode版本函数结尾以W表示。Windows会如下定义：   
#ifdef UNICODE   
#define CreateWindowEx CreateWindowExW   
#else   
#define CreateWindowEx CreateWindowExA   
#endif // !UNICODE   
7. 如何表示Unicode字符串常量？   
字符集 实例   
ANSI “string”   
Unicode L“string”   
ANSI/Unicode T(“string”)或\_TEXT(“string”)if( szError[0] == \_TEXT(‘J’) ){ }   
8. 为什么应当尽量使用操作系统函数？   
这将有助于稍稍提高应用程序的运行性能，因为操作系统字符串函数常常被大型应用程序比如操作系统的外壳进程Explorer.exe所使用。由于这些函数使用得很多，因此，在应用程序运行时，它们可能已经被装入RAM。   
如：StrCat，StrChr，StrCmp和StrCpy等。   
9. 如何编写符合ANSI和Unicode的应用程序？   
（1） 将文本串视为字符数组，而不是chars数组或字节数组。   
（2） 将通用数据类型（如TCHAR和PTSTR）用于文本字符和字符串。   
（3） 将显式数据类型（如BYTE和PBYTE）用于字节、字节指针和数据缓存。   
（4） 将TEXT宏用于原义字符和字符串。   
（5） 执行全局性替换（例如用PTSTR替换PSTR）。   
（6） 修改字符串运算问题。例如函数通常希望在字符中传递一个缓存的大小，而不是字节。这意味着不应该传递sizeof(szBuffer),而应该传递（sizeof(szBuffer)/sizeof(TCHAR)。另外，如果需要为字符串分配一个内存块，并且拥有该字符串中的字符数目，那么请记住要按字节来分配内存。这就是说，应该调用malloc(nCharacters \*sizeof(TCHAR)),而不是调用malloc(nCharacters)。   
10. 如何对字符串进行有选择的比较？   
通过调用CompareString来实现。   
标志 含义   
NORM\_IGNORECASE 忽略字母的大小写   
NORM\_IGNOREKANATYPE 不区分平假名与片假名字符   
NORM\_IGNORENONSPACE 忽略无间隔字符   
NORM\_IGNORESYMBOLS 忽略符号   
NORM\_IGNOREWIDTH 不区分单字节字符与作为双字节字符的同一个字符   
SORT\_STRINGSORT 将标点符号作为普通符号来处理   
11. 如何判断一个文本文件是ANSI还是Unicode？   
判断如果文本文件的开头两个字节是0xFF和0xFE，那么就是Unicode，否则是ANSI。   
12. 如何判断一段字符串是ANSI还是Unicode？   
用IsTextUnicode进行判断。IsTextUnicode使用一系列统计方法和定性方法，以便猜测缓存的内容。由于这不是一种确切的科学方法，因此 IsTextUnicode有可能返回不正确的结果。   
13. 如何在Unicode与ANSI之间转换字符串？   
Windows函数MultiByteToWideChar用于将多字节字符串转换成宽字符串；函数WideCharToMultiByte将宽字符串转换成等价的多字节字符串。