**LPSTR、LPCSTR、LPTSTR和LPCTSTR的意义及区别**

2009-09-28 17:09:58| 分类： [默认分类](http://kingarthurjs.blog.163.com/blog/#m=0&t=1&c=fks_087071082082087064081081084095085087087069081087095067081)|字号 订阅

1、ANSI（即MBCS）：为多字节字符集，它是不定长表示世界文字的编码方式。ANSI表示英文字母时就和

ASCII一样，但表示其他文字时就需要用多字节。

2、Unicode：用两个字节表示一个字符的编码方式。比如字符'A'在ASCII下面用一个字节表示，而在

Unicode下面用两个字节表示，其中高字节用“0”填充；函数'程'在ASCII下面用两个字节表示，而在

Unicode下面也是用两个字节表示。Unicode的用处就是定长表示世界文字，据统计，用两个字节可以编码

现存的所有文字而没有二义。

3、Windows下的程序设计可以支持ANSI和Unicode两种编码方法的字符串，具体使用哪种就要看定义了

MBCS宏还是Unicode宏。MBCS宏对应的字符串指针为LPSTR（即char\*），Unicode对应的指针为

LPWSTR（即unsigned char\*）。为了写程序的方便，微软定义了类型LPTSTR，在MBCS下它表示char\*，

在Unicode下它表示unsigned char\*，这就可以重定义一个宏进行不同字符集的转换了。

4、关系

LPSTR：指向一个字符串的32位指针，每个字符占1个字节。

LPCSTR：指向一个常量字符串的32位指针，每个字符占1个字节。

LPTSTR：指向一个字符串的32位指针，每个字符可能占1个字节或2个字节。

LPCTSTR：指向一个常量字符串的32位指针，每个字符可能占1个字节或2个字节。

5、Windows使用两种字符集ANSI和Unicode，前者在处理英文字符时使用单字节方式，在处理中文字符时

使用双字节方式。后者不管是英文字符还是中文字符都是采用双字节方式表示。Windows NT的所有与字符

有关的函数都提供了两种方式的版本，而Windows 9x只支持ANSI方式。\_T一般同字符常量相关，如\_T("你

好！");。如果编码方式为ANSI，\_T不起任何作用；如果编码方式为Unicode，编译器会把"你好！"以

Unicode方式进行保存。\_T和\_L的区别在于，\_L不管你是以什么方式编译，一律以Unicode方法保存。

6、L表示字符串资源采用Unicode编码方式，示例如下：

wchar\_t Str[]=L"Hello World!"; //每个字符都用2个字节来存储

7、\_T是一个适配宏。当定义了\_UNICODE时，\_T和L相同；否则，\_T的字符串采用ANSI编码方式。示例如

下：

LPTSTR lptStr=new TCHAR[32];

TCHAR\* szBuf=\_T("Hello");

以上两个语句，无论是在ANSI编码方式，还是在Unicode编码方式下都是正确的。

8、微软推荐使用相匹配的字符串函数。例如，在处理LPTSTR或LPCTSTR的时候，应该使用\_tcslen来替

代strlen函数。否则，在Unicode编码方式下，strlen不能处理wchar\_t\*的字符串。

9、T是一个非常重要的符号（TCHAR、LPTSTR、LPCTSTR、\_T()和\_TEXT()等），它表示使用一个中间

类型，既不明确表示使用MBCS，也不明确表示使用Unicode。到底使用哪种字符集，在编译的时候才决

定。

10、CString类型到LPTSTR类型的转换

CString path1;

LPTSTR path2=path1.GetBuffer(path1.GetLenght());

**规范化编程：ANSI和UNICODE的使用**

2007-01-29 23:01:48| 分类： [软件技术--转载](http://25110202.blog.163.com/blog/#m=0&t=1&c=fks_082070087085088064087074095075092084087074087)|字号 订阅

到底什么是ANSI，什么是UNICODE呢？其实这是两种不同的编码方式标准，ANSI中的字符采用8bit，而UNICODE中的字符采用16bit。8bit的ANSI编码只能表示256种字符，表示26个英文字母是绰绰有余的，但是表示汉字，韩国语，日语等有着成千上万个字符的非西方字符肯定就不够了，正是如此才引入了UNICODE标准。

在软件开发中，特别是使用C语言的一些有关字符串处理的函数，ANSI和UNICODE是区分是用的，那么ANSI类型的字符和UNICODE类型的字符如何定义，如何使用呢？ANSI和UNICODE又如何转换呢？

一．定义部分：

ANSI： **char** str[1024]; 可用字符串处理函数：strcpy( ), strcat( ), strlen( )等等。UNICODE：**wchar\_t**str[1024]；可用字符串处理函数

二．可用函数：

ANSI：即 **char，** 可用字符串处理函数：**strcat( ),strcpy( ), strlen( )**等以**str**打头的函数。

UNICODE：即**wchar\_t** 可用字符串处理函数：**wcscat(),wcscpy(),wcslen()**等以**wcs**打头的函数。

三．系统支持

Windows 98 ：只支持ANSI。

Windows 2k ：既支持ANSI又支持UNICODE。

Windows CE ：只支持UNICODE。

说明

1． 在COM里面只支持UNICODE。

2． Windows 2000整个OS系统都是基于UNICODE的，为此在windows 2000下使用ANSI是需要付出代价的，虽然在编码上不用任何的转换，但是这种转化是隐藏的，是占用系统资源的（CPU，内存）。

3． 在Windows 98下必须使用UNICODE，则需要自己手动的编码切换。

三．如何区分：

在我们软件开发中往往需要即支持ANSI又支持UNICODE，不可能在要求类型转换的时候，重新改变字符串的类型，和使用于字符串上的操作函数。为此, 标准的C 运行期库和Windows 提供了宏定义的方式。

在C语言里面提供了 \_UNICODE宏（有下划线），在Windows里面提供了UNICODE宏（无下划线），只要定了\_UNICODE宏和UNICODE宏，系统就会自动切换到UNICODE版本，否则，系统按照ANSI的方式进行编译和运行。

只定义了宏并不能实现自动的转换，他还需要一系列的字符定义支持。

1． TCHAR

如果定义了UNICODE宏则TCHAR被定义为wchar\_t。

typedef wchar\_t TCHAR;

否则TCHAR被定义为char

typedef char TCHAR;

2． LPTSTR

如果定义了UNICODE宏则LPTSTR被定义为LPWSTR。

typedef LPTSTR LPWSTR;

否则TCHAR被定义为char

typedef LPTSTR LPSTR;

**UNICODE 编程入门**

2009-12-18 17:29:20| 分类： vc++ | 标签：编程 unicode |字号 订阅

简介

如果你编写的程序是针对非英语国家的用户，如中国、日本、东欧和中东地区，那么你一定要熟悉 UNICODE 字符集。尤其是用 Visual C++/MFC 编写针对上述国家和地区的用户的程序时，如果你想让自己的应用程序得到更广泛的用户，那么必须考虑代码 UNICODE 的兼容性，也就是说它既在 ASCII 模式下运行 ，也能在UNICODE 模式下运行。本文将介绍 UNICODE 的一些基本编程知识，澄清很多人（包括我自己）在这个问题上存在的模糊认识。对于任何使用 Visual C++ 和/或 MFC 编程的人来说，这篇文章肯定值得一读。

UNICODE到底是什么？

UNICODE 是目前用来解决 ASCII 码 256 个字符限制问题的一种比较流行的解决方案。大家知道，ASCII 字符集只有256个字符，用 0-255 之间的数字来表示。包括大小写字母、数字以及少数特殊字符；如标点符号、货币符号等。对于大多数拉丁语言来说，这些字符已经够用。但是，许多亚洲和东方语 言所用的字符远远不止256个字符。有些超过千个。人们为了突破 ASCII 码字符数的限制，试图用一种简单的方法来针对超过256个字符的语言编写计算机程序。于是 UNICODE 应运而生。UNICODE 通过用双字节来表示一个字符，从而在更大范围内将数字代码映射到多种语言的字符集。

Visual C++的解决方案

作为软件开发人员，如何熟练有效地使用 UNICODE 呢？如果你正在用 Visual C++ 编写程序，UNICODE 兼容性意味着你的程序是否具有国际化特征，也就是说你的应用程序是针对本地市场还是国际市场。一旦你作出了决定，那么就得在代码中实现具体细节。好在 Visual C++ 提供了很多内建功能来支持 UNICODE，在创建工程时就可以利用 Visual C++ 提供的这些功能。在产生应用程序框架代码之前，AppWizard 允许开发人员决定是否支持 UNICODE。Win32 SDK 包含有一些数据类型遵循 UNICODE 编码规则，MFC 以宏的形式提供了将一般文本转换成 UNICODE 数据类型的途径。开发人员只需要稍微改变一下编写代码的习惯便可以轻松编写支持 UNICODE 的应用。

字符串

C 程序员一般是用 char 关键字象下面这样来声明一个字符串数组：

char str[100];

象下面这样声明函数原形：

void strcpy( char \*out, char \*in );

为了将上面的声明改成支持双字节的 UNICODE 字符集，可以用下面的方法：

wchar\_t str[100];

或者

void wcscpy( wchar\_t \*out, wchar\_t \*in );

此外，微软还提供一种通过预处理指令来实现 UNICODE。每当用 Visual C++ 创建新工程时，只要确定是否支持另外一种字符集，则 AppWizard 将会在头文件中插入预处理指令。这些指令告诉编译器程序想要支持何种字符集。这样在使用VC++提供的通用数据类型时，编译器将用相应的数据类型把通用数据类型替换成所需要支持的字符集。这样很容易将代码重新编译成支持其它字符集的程序。

为了在 Visual C++ 6.0 中激活 UNICODE 标准，可以这样做：打开工程文件后，从主菜单中选择“Project | Settings”打开工程设置对话框 => 然后选择“C/C++”标签 => 在“Preprocessor definitions”编辑框中添加 UNICODE 或者 \_UNICODE 预处理宏指令。

注意这里的 UNICODE 和 \_UNICODE 有什么区别呢？前者没有下划线，专门用于 Windows 头文件；后者有一个前缀下划线，专门用于 C 运行时头文件。

在代码中，凡是用关键字 char 的地方都用 TCHAR 取代；凡是用 char \* 的地方都用 LPTSTR 取代；凡是定义在双引号中的字符串常量（如"VCKBASE Online Journal"）都用 TEXT 宏重写：

TEXT("VCKBASE Online Journal");

TEXT 宏的主要作用是当定义了 UNICODE/\_UNICODE 预处理指令时，字符串被标志为双字节字符串，否则字符串被标示为 ANSI 字符串。TEXT 的定义如下：

TEXT(

LPTSTR string // ANSI 或者 Unicode 字符串

);

参数 string 为字符串指针，指向被解释的 Unicode 或者 ANSI 字符串

在文档中 微软提供了包括通用类型在内的几种数据类型都与 ASCII 和 UNICODE兼容。这一点可以参考微软在线文档有关“通用数据类型和数据类型”的章节。

例子代码

下面通过一些简单的例子来进一步探讨 UNICODE 编程。

使用 ASCII 字符集的“Hello, World”：

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 用 MFC 实现的"Hello World!" 代码

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//hello.cpp

#include <afxwin.h>

// Declare the application class

class CHelloApp : public CWinApp

{

public:

virtual BOOL InitInstance();

};

// Create an instance of the application class

CHelloApp HelloApp;

// Declare the main window class

class CHelloWindow : public CFrameWnd

{

CStatic\* cs;

public:

CHelloWindow();

};

// The InitInstance function is called each

// time the application first executes.

BOOL CHelloApp::InitInstance()

{

m\_pMainWnd = new CHelloWindow();

m\_pMainWnd->ShowWindow(m\_nCmdShow);

m\_pMainWnd->UpdateWindow();

return TRUE;

}

// The constructor for the window class

CHelloWindow::CHelloWindow()

{

// Create the window itself

Create(NULL, "Hello World!", WS\_OVERLAPPEDWINDOW,

CRect(0,0,200,200));

// Create a static label

cs = new CStatic();

cs->Create("hello world", WS\_CHILD|WS\_VISIBLE|SS\_CENTER,

CRect(50,80,150,150), this);

}

修改上面的代码使之支持 UNICODE 字符集，串常量必须要改成对应的 UNICODE 字符。方法是对串常量使用TEXT 宏。这个宏将告诉预处理器检查使用什么样的字符标准：

// The constructor for the window class

CHelloWindow::CHelloWindow()

{

// Create the window itself

Create(NULL, TEXT("Hello World!"), WS\_OVERLAPPEDWINDOW,

CRect(0,0,200,200));

// Create a static label

cs = new CStatic();

cs->Create( TEXT("hello world!"), WS\_CHILD|WS\_VISIBLE|SS\_CENTER,

CRect(50,80,150,150), this);

}

当预处理器碰到通用数据类型，它便检查 AFXWIN.H 头文件的 \_UNICODE 定义。然后根据 UNICODE 定义插入相应的的数据类型。

下面的这个例子使用 Win32 API 函数和通用数据类型设置 C 盘的卷标。

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 设置 C 盘的卷标

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// drvsvl.cpp

#include <windows.h>

#include <iostream.h>

void main()

{

BOOL success;

char volumeName[MAX\_PATH];

cout << "输入新的 C 盘卷标：";

cin >> volumeName;

success = SetVolumeLabel("c:\\", volumeName);

if (success)

cout << "成功\n";

else

cout << "错误代码：" << GetLastError() << endl;

}

通过使用 TCHAR 数据类型，将这段代码最上面的字符数组声明为两个字节的字符。TEXT 宏再次被用于字符串常量：

void main()

{

BOOL success;

TCHAR volumeName[MAX\_PATH];

cout << TEXT("输入新的 C 盘卷标： ");

cin >> volumeName;

success = SetVolumeLabel(TEXT("c:\\" ), volumeName);

if (success)

cout << TEXT("成功\n");

else

cout << TEXT("错误代码：") << GetLastError() << endl;

}

Visual C++ 中的通用数据类型

Visual C++ 提供了几种 MFC 专用的数据类型用于创建具有国际化特性的应用程序。这些定义很通用，完全可以在 UNICODE、ASCII、DBCS (双字节字符集) 和 MBCS (多字节字符集)。由于篇幅所限，本文不打算涉及所有上面提到的这些字符集。有关它们的详细资料请参考相关资料。MFC 提供了一种透明的方式来实现这些字符集。通用数据类型的映射到哪个字符集以及映射方式是根据工程的设置决定的，默认值为 ASCII 模式，其它几个可选项是 MBCS、DBCS 或者 UNICODE。本文主要讨论 UNICODE，所以下表中只列出了 ASCII 与 UNICODE 字符之间的映射关系：

表一：

通用 MFC 数据类型

映射到 ASCII

映射到 UNICODE

注释

\_TCHAR

char

wchar\_t

\_TCHAR 是一个映射宏，当定义 UNICODE 时，该数据类型映射到 wchar\_t，如果没有定义 UNICODE，那么它映射到 char。

\_T 或 \_TEXT

char 常量字符串

wchar\_t 常量字符串

功能与宏相同，在 ASCII 模式下，它们被忽略，也就是说被预处理器删除掉，但是如果定义了UNICODE, 则它们会将常量字符串转换成等价的 UNICODE 。

LPTSTR

char\*, LPSTR(Win32)

wchar\_t\*

可移植的32位字符串指针。它将字符类型映射到工程设置的类型。

LPCTSTR

const char\*, LPCSTR(Win32)

const wchar\_t\*

可移植的32位常量字符串指针。它将字符类型常量映射到工程设置的类型。

使用表一中列出的通用数据类型，开发人员可以保证所创建的工程始终是针对一种字符集，这些通用数据类型就相当于占位符，在编译时被特定的字节所替代，使得应用程序在 ASCII 和 UNICODE 模式下都能运行。但是，有一点要特别注意，那就是上述的通用数据类型为微软专有，与 ANSI 标准并不兼容。有关微软提供的这些通用数据类型详细描述请参考 MSDN 库文档。

有关技术注释

为了成功编译支持 UNICODE 的 MFC 程序，必须使用 MFC 的 UNICODE 版本库。该库在定制安装Visual C++ 时是个可选安装项。

有一点很重要：那就是不使用 UNICODE 标准在外观上并不影响程序的执行。也就是说，上面提到过的代码不管设没设置 \_UNICODE 生成选项，最终都能生成正常运行的程序。当开发人员使用多个版本的Win32 API函数时才会出现问题。

在使用多个版本的 Win32 API函数（任何有字符或字符串作为参数的 Win32 API函数）时，编译器根据是否设置 \_UNICODE 指令来决定调用正确的函数。如果没有定义\_UNICODE，那么编译器将默认调用 ASCII 版本函数。

结束语

综上所述可以看到，编译 UNICODE 版本的程序并不难。只是在编写代码时记住函数调用上些微的变化。微软为此提供的扩展是开发人员能够以透明的方式选择所用的字符集，为应用软件的国际化打开了方便之门。

Jeffrey Richter 在他的《Windows 核心编程》（机械工业出版社-王建华、张焕生、侯丽坤等译）一书中专门用一章讨论了 UNICODE。翻译得也不错。有兴趣的朋友不妨找来看看。

参考资料：

<http://www.vckbase.com/document/viewdoc/?id=642>

**Unicode宽字符转化**

2009-11-06 19:10:17| 分类： [VC/C++](http://09105106.blog.163.com/blog/#m=0&t=1&c=fks_087069093086088075085081094095086082089064081081094)|字号 订阅

// 把宽字符转化为ANSI字符串  
WideCharToMultiByte( CP\_ACP, 0, lpcwStr, -1, lpStr, sizeof(lpStr), NULL, NULL );

// 把ANSI字符串转化为宽字符  
MultiByteToWideChar( CP\_ACP, 0, lpcStr, -1, lpwStr, sizeof(lpwStr) );

int <— \_ttoi <— CString  
double <— \_tcstod <— CString  
CString <— .Format <— int  
CString <— .Format <— double

为支持Unicode做准备，除个别必须使用 char 或 char\* 的地方外，  
其它地方请依照如下标准：

1.字符串类型尽量用CString  
2.字符类型使用 TCHAR  
3.字符串数组用 TCHAR[]  
4.字符串指针用 TCHAR\*  
5.常量字符串指针用 const TCHAR\*  
6.字符串常量和字符常量加宏 \_T

特别的：   
1>CString::Format(\_T("..."),...)//要加\_T  
2>LPTSTR 等等被多次嵌套定义过的宏尽量少用，用基础类型 const TCHAR\*(或TCHAR\*)替代;  
3>为TCHAR\*类型指针分配 N-1 个字符的空间: TCHAR\* pbuffer = new TCHAR[ N\*sizeof(TCHAR) ];

常用字符串函数，须使用支持Unicode/Ansi的宏函数替换前者：

Ansi字符串函数 Unicode/Ansi宏函数 功能说明   
 strcpy \_tcscpy 字符串拷贝   
 strcat \_tcscat 字符串连接  
 strlen \_tcslen 求字符串长度   
 strcmp \_tcscmp 字符串比较   
 atof \_tcstod (注) 字符串转换为double   
 atoi \_ttoi 字符串转换为int   
注：上表中 \_tcstod 和 atof用法不同，不能直接替换。

LPCTSTR 使用UELPCTSTR代替  
std::string 使用UeStdString代替  
这个需要加ShareHeader.h头文件  
#ifdef UNICODE  
typedef std::wstring UeStdString;  
#define UELPCSTR LPCWSTR  
#else  
typedef std::string UeStdString;  
#define UELPCSTR LPCTSTR  
#endif

**VC中的字符串类型和处理函数**

2009-05-17 16:10:10 分类： [VC++](http://fbcmd.blog.163.com/blog/#m=0&t=1&c=fks_087071083080083066086095087095085087086070086087094070092)

VC中的字符串类型和处理函数是一个比较复杂的话题，我准备写一篇文章详细地讲述这个问题。今天的内容仅限于处理字符串的C函数。

这类函数共有四族，以取字符串长度的函数为例，四族函数如下：

strlen str族  
wcslen wcs族  
\_mbslen \_mbs族  
\_tcslen \_tcs族

其中，前两个族的函数是C语言的标准库函数，以下划线开始的两族函数是VC中特有的函数，即微软的扩展函数。

这四族函数最大的区别是它们所用的字符集不同，或者说它们的参数类型不同。

str族是我们都比较熟悉的，它的参数类型是char\*，这一族函数均以字节为单位进行处理，一个英文字符占一个字节，一个汉字占两个字节，因此strlen("中文ABC")的结果是7，字符串在内存中实际占用7个字节。

再说说wcslen，它的参数类型是wchar\_t\*，wchar\_t称为宽字符，或者UNICODE字符，是用两个字节表示一个字符。UNICODE字符串都要在前面加上一个L前缀，因此，\_wcslen(L"中文ABC")的结果是5，但字符串在内存中占10个字节。

\_mbslen似乎很少用，它的参数类型是unsigned char\*，\_mbs族函数是为了处理汉字字符专门引入的，它以实际的字符为单位而不是以字节为单位进行处理，一个汉字是一个字符，因此\_mbslen("中文ABC")的结果是5，但字符串在内存中占7个字节。

\_tcslen称为自适应函数，它的参数类型是TCHAR，TCHAR是一个自适应类型。在UNICODE环境下\_tcslen等效于wcslen，TCHAR等效于wchar\_t，在非UNICODE环境下，\_tcslen等效于strlen，TCHAR等效于char。因此，\_tcslen(\_T("中文ABC"))的结果可能是7或5，字符串本身在内存中可能占7个字节，也可能占10个字节，依环境而定。  
  
综上所述，如果你的程序既要制作成UNICODE版，又要制作成非UNICODE版，最好使用\_tcs族函数和TCHAR类型，并且每个字符串常量最好用\_T()括起来。