这是啥都没有的封面

目录

[字符串中的一些宏定义和变量类型字母意义 1](#_Toc489289388)

[字母含义 1](#_Toc489289389)

[宏定义含义 1](#_Toc489289390)

[其他注释 1](#_Toc489289391)

[<1> 1](#_Toc489289392)

[<2> 2](#_Toc489289393)

[字符类型的相互转化 3](#_Toc489289394)

[std::string,std::wstring 4](#_Toc489289395)

[定义 4](#_Toc489289396)

[其他 4](#_Toc489289397)

[CString 5](#_Toc489289398)

[定义 5](#_Toc489289399)

[其他 5](#_Toc489289400)

[BSTR 6](#_Toc489289401)

[定义 6](#_Toc489289402)

[其他备注 6](#_Toc489289403)

[CComBSTR 7](#_Toc489289404)

[定义 7](#_Toc489289405)

[其他备注 7](#_Toc489289406)

[\_bstr\_t 8](#_Toc489289407)

[定义 8](#_Toc489289408)

[其他 8](#_Toc489289409)

[PCTSTR, LPCTSTR（PCUTSTR, LPCUTSTR） 9](#_Toc489289410)

[定义 9](#_Toc489289411)

[其他： 9](#_Toc489289412)

[PTSTR, LPTSTR（LP,PUTSTR, LPUTSTR） 10](#_Toc489289413)

[定义 10](#_Toc489289414)

[其他 10](#_Toc489289415)

[其他字符类型 11](#_Toc489289416)

# 字符串中的一些宏定义和变量类型字母意义

c++中的大部分字符串类型变量定义中都包含根据设置改变单双字节的定义

基础部分基本为：

#ifdef Unicode（或者其他类似单双字节的全局定义）

typedef WCHAR xxx

#else

typedef char xxx

#endif

## 字母含义

L表示long指针 这是为了兼容Windows 3.1等16位操作系统遗留下来的，在win32中以及其他的32位操作系统中， long指针和near指针及far修饰符都是为了兼容的作用。没有实际意义。

P表示这是一个指针

C表示是一个const常量

T表示在Win32环境中， 有一个\_T宏（随单双字节环境改变）

U表示UNALIGNED 好像是针对某些情况需要对齐内存？（没有见过实际运用）

W表示多字节字符

STR表示这个变量是一个字符串

## 宏定义含义

\_Null\_terminated\_ Null-terminated string 其实就是着的是“以'\0'结尾的字符串”。

basic\_string STL只有一个字符串类，basic\_string

char\_traits 是base\_string的一个模板参数，它主要负责关于字符的属性和方法，譬如 eq、lt、compare、find 这些，比较字符大小，查找字符等等

## 其他注释

### <1>

**winNT.h与wType.h两个头**中有很多变量定义冲突，当两者冲突时

**MFC中采用的大部分是winNT.h**

### <2>

OLECHAR在**wtypes.h和afxwin,h**中有不同的定义

MFC默认包含stdafx.h，以afxwin.h中的定义为准

**afxwin,h中**

typedef WCHAR OLECHAR;

**wtypes.h中**

#if defined(\_WIN32) && !defined(OLE2ANSI)

typedef WCHAR OLECHAR;

#else

typedef char OLECHAR;

#endif

**（所以在非MFC程序中BSTR可以被定义为单字节字符串数据类型）**

# 字符类型的相互转化

注：以下转化在MFC/Unicode环境下测试通过

CComBSTR bstrCom;

BSTR bstr;

\_bstr\_t bstrt;

LPCTSTR pcstr;

LPTSTR pstr;

string str;

CString cstr;

//这里注意，由于BSTR是字符串指针，不能直接使用，需要分配内存

bstr = ::SysAllocString(\_T("test"));

// CComBSTR 和 BSTR 之间可以直接相互转换

bstrCom = bstr;

//\_bstr\_t 到 CComBSTR 的转化需要用 BSTR 作为中介

bstrt = bstr;

bstrCom = (BSTR)bstrt;

bstrt = bstrCom;

//除了LPCTSTR 类型由于const限定符，与其他类型之间需要强制类型转化，其他类型都可以直接互相赋值

pstr = bstr;

pstr = bstrt;

pcstr = pstr;

pstr = (LPTSTR)pcstr;

bstrCom = (CComBSTR)pcstr;

// string与其他类型相互赋值时需要中介和强转，好像是因为特殊的包装方式？

只有\_bstr\_t类型可以直接与其相互赋值

str = (\_bstr\_t)bstr;

str = (\_bstr\_t)pstr;

str = (\_bstr\_t)cstr;

bstr = (LPTSTR)str.c\_str();

bstr = \_bstr\_t(str.c\_str());

cstr = (LPTSTR)str.c\_str();

# std::string,std::wstring

基于basic\_string包装的c++字符串类

## 定义

c++标准头文件xstring中

class basic\_string

: public \_String\_alloc<\_String\_base\_types<\_Elem, \_Alloc> >

typedef basic\_string<char, char\_traits<char>, allocator<char> >

string;

typedef basic\_string<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >

wstring;

## 其他

这是一个c++基础类，可以脱离MFC单独使用

# CString

cstring是MFC特有的字符串数据类型，包装了大量字符串操作方法

## 定义

cstringt.h中

class CStringT :

public CSimpleStringT< BaseType, \_CSTRING\_IMPL\_::\_MFCDLLTraitsCheck<BaseType, StringTraits>::c\_bIsMFCDLLTraits >

MFC头文件afxstr.h中

#if defined(\_AFXDLL)

typedef ATL::CStringT< wchar\_t, StrTraitMFC\_DLL< wchar\_t > > CStringW;

typedef ATL::CStringT< char, StrTraitMFC\_DLL< char > > CStringA;

typedef ATL::CStringT< TCHAR, StrTraitMFC\_DLL< TCHAR > > CString;

#else

typedef ATL::CStringT< wchar\_t, StrTraitMFC< wchar\_t > > CStringW;

typedef ATL::CStringT< char, StrTraitMFC< char > > CStringA;

typedef ATL::CStringT< TCHAR, StrTraitMFC< TCHAR > > CString;

#endif

## 其他

这个数据类型仅能在MFC工程中使用

# BSTR

"**Basic STRing**"的简称，微软在COM/OLE中定义的标准字符串数据类型。对于C++，

## 定义

Windows头文件**wtypes.h**中

typedef wchar\_t WCHAR;

typedef WCHAR OLECHAR;

typedef OLECHAR \_\_RPC\_FAR \*BSTR;

## 其他备注

这是一个双字节的字符类型，一个字符站两个字节

BSTR的前4字节是一个表示字符串长度的前缀。

BSTR长度域的值是字符串的字节数，并且不包括0结束符。

对于C++程序，如何实现BSTR和OLECHAR的交换？答案是COM提供了两个BSTR分配用的API：SysAllocString / SysReallocString。函数返回的指针指向BSTR的第一个字符，而不是BSTR在内存的第一个字节。

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

0a000000 | H | E | L | L | O | /0|

^

BSTR

**这个类型需要手动释放，调用函数：SysFreeString**

# CComBSTR

ATL 类 CComBSTR 提供对 BSTR数据类型的包装。

## 定义

**atlcomcli.h**

class CComBSTR

{

public:

BSTR m\_str;

.

.

.

}

## 其他备注

CComBSTR 类提供了一大批带 ANSI 或 Unicode 字符串作为参数的成员函数 （构造函数、 赋值运算符和比较运算符）。这些函数的 ANSI 版本比 Unicode版本 效率较低，因为临时 Unicode 字符串通常内部创建。为了提高效率，尽可能使用 Unicode 版本。

**CComBSTR在超出作用域时会自动释放**，因此不建议直接使用BSTR，建议使用CComBSTR作为替代

# \_bstr\_t

\_bstr\_t是c++为COM定义的一个BSTR数据类型的包装，类似与CComBSTR，自动管理内存并在超出作用域时释放。

这个数据类型是c++提供的BSTR的扩展，不需要包含额外的ATL头文件

## 定义

comutil.h

class \_bstr\_t {

private:

BSTR m\_wstr;

mutable char\* m\_str;

unsigned long m\_RefCount;

.

.

.

}

## 其他

同样的，由于该类型提供自动释放，所以也是一个BSTR的替代选择

# PCTSTR, LPCTSTR（PCUTSTR, LPCUTSTR）

括号中的两种一般情况不会被用到

L表示long指针 这是为了兼容Windows 3.1等16位操作系统遗留下来的，在win32中以及其他的32位操作系统中， long指针和near指针及far修饰符都是为了兼容的作用。没有实际意义。

P表示这是一个指针

C表示是一个const常量

T表示在Win32环境中， 有一个\_T宏

STR表示这个变量是一个字符串

## 定义

Windows头文件**winnt.h**中

#ifdef UNICODE

typedef wchar\_t WCHAR;

typedef \_Null\_terminated\_ CONST WCHAR \*LPCWSTR, \*PCWSTR;

typedef LPCWSTR PCTSTR, LPCTSTR;

#else

typedef \_Null\_terminated\_ CONST CHAR \*LPCSTR, \*PCSTR;

typedef LPCSTR PCTSTR, LPCTSTR, PCUTSTR, LPCUTSTR;

#endif

Windows头文件**wtypes.h**中

typedef const TCHAR \*LPCTSTR;

## 其他：

从定义中可以看到这是一个字符串类型的常量指针，由宏定义控制其单双字节

# PTSTR, LPTSTR（LP,PUTSTR, LPUTSTR）

括号中的两种一般情况不会被用到

L表示long指针 这是为了兼容Windows 3.1等16位操作系统遗留下来的，在win32中以及其他的32位操作系统中， long指针和near指针及far修饰符都是为了兼容的作用。没有实际意义。

P表示这是一个指针

T表示在Win32环境中， 有一个\_T宏

STR表示这个变量是一个字符串

## 定义

Windows头文件**winnt.h**中

#ifdef UNICODE

typedef wchar\_t WCHAR;

typedef \_Null\_terminated\_ WCHAR \*NWPSTR, \*LPWSTR, \*PWSTR;

typedef LPWSTR PTSTR, LPTSTR;

typedef LPWSTR LP;

#else

typedef \_Null\_terminated\_ CHAR \*NPSTR, \*LPSTR, \*PSTR;

typedef LPSTR PTSTR, LPTSTR, PUTSTR, LPUTSTR;

#endif

Windows头文件**wtypes.h**中

typedef TCHAR \*LPTSTR;

## 其他

与上一条相近，这是一个字符串类型的指针，不同的是这是一个变量而非常量

与上一条可以互相强制类型转化

# 其他字符类型

这些类型一般不直接用到，除了wchar可能会用在文件流的读写中

LPCWSTR

LPCSTR

LPWSTR

LPSTR

TCHAR

wchar\_t

WCHAR