**按行读取ANSI、UNICODE 、UNICODE big endian、UTF-8四种文本文**

2016年06月20日 16:35:59 [龙行天下之Sky](https://me.csdn.net/sky101010ws) 阅读数：2871 标签： [mfc](http://so.csdn.net/so/search/s.do?q=mfc&t=blog" \t "_blank)[unicode](http://so.csdn.net/so/search/s.do?q=unicode&t=blog)更多

个人分类： [学习点滴](https://blog.csdn.net/sky101010ws/article/category/1432884)

https://blog.csdn.net/sky101010ws/article/details/51721758

代码简介

**1.问题提出**  
MFC提供的文件类CStdioFile，其中一个函数ReadString实现了文件的按行读取，但是不能满足不同类型的文本文件的按行读取，为了解决这一问题，笔者**初步研究**了一些编码知识，参考了网上的一些资料，实现了CStdioFile类的扩展类CStdioFileEx,完成了常见文本文件的按行读取（注明：不包括DOC、PDF等其他形式的文档）.  
在此对网上分享编码经验的网友表示感谢，同时由于我编写的类还未经过严格测试，如有错误或方法过于复杂敬请各位指正。   
**2.问题解决**  
(1)四种常见文本文件编码方式研究  
**ANSI、UNICODE 、UNICODE big endian、UTF-8**四种格式编码存在差别，简要介绍如下:  
ANSI编码:  
无文件头(文件编码开头标志性字节)      
ANSI编码字母数字占一个字节，汉字占两个字节，  
回车换行符 单字节   十六进制表示为0d  0a  
  
UNICODE编码:  
文件头，十六进制表示为FF FE  
每一个字符都用两个字节编码  
回车换行符  双字节   000d  000a  
  
Unicode big endian编码:  
文件头十六进制表示为FE FF ，  
后面编码是把字符的高位放在前面，低位放在后面，正好和Unicode编码颠倒。  
回车换行符,双字节,十六进制表示为0d00  0a00  
  
UTF-8 编码:  
文件头，十六进制表示为EF BB BF。  
UTF-8是Unicode的一种变长字符编码，数字、字母、回车、换行都用一个字节表示,汉字占3个字节.  
回车换行符，单字节，十六进制表示为0d 0a  
  
以中文"你好"二字为例，各种类型的编码对应的十六进制格式(可由EditPlus查看)如下图所示:  
  
由此可见上述的探讨是正确的。  
(2)按行读取上述四种格式文本文件的解决方案  
针对不同文件编码的特点，通过先检测文件头判断文件编码类型，然后根据文件类型分别调用不同的读取函数实现文件的按行读取。按行读取过程如下图所示:  
  
实现过程中，编写CStdioFileEx类，该类继承自CStdioFile类，覆盖了CStdioFile类的BOOL ReadString(CString& rString)方法，从而实现了文件按行读取。

代码片段

// StdioFileEx.h: interface for the CStdioFileEx class.

//

//////////////////////////////////////////////////////////////////////

#if !defined(AFX\_STDIOFILEEX\_H\_\_C1F1F96B\_9417\_4388\_8D24\_892EDFA2A616\_\_INCLUDED\_)

#define AFX\_STDIOFILEEX\_H\_\_C1F1F96B\_9417\_4388\_8D24\_892EDFA2A616\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

// --------------------------------------------------------------------------------------------

//程序用途：按行读取常见(包括ANSI、UNICODE、UNICODE big endian、UTF-8)格式的文本文件

//程序作者：湖北师范学院计算机科学与技术学院 王定桥

//核心算法：CStdioFileEx继承自CStdioFile， 覆盖CStdioFile的 BOOL ReadString(CString& rString)方法,

// 根据不同文件编码特征，寻找文件回车换行符判断读取行结束，文件结束符判断文件结束

// 检测不同文件编码头部，获取文件类型后调用不同的读取函数

//测试结果:在Windows7 VC6.0环境下测试上述四种格式的txt文件通过

//尚未完成:未重载CStdioFile的 virtual LPTSTR ReadString( LPTSTR lpsz, UINT nMax )方法

// 未完成WriteString方法,未在VC UNICODE 环境下的测试

//制作时间：2012-04-19

//代码版权:代码公开供学习交流使用 欢迎指正错误 改善算法

// --------------------------------------------------------------------------------------------

#include "stdafx.h"

//文本文件类型枚举值

typedef enum TextCodeType

{

UTF8=0,

UNICODE =1,

UNICODEBIGENDIAN=2,

ANSI=3,

FILEERROR=4

}TextCode;

class CStdioFileEx :public CStdioFile

{

public:

CStdioFileEx();

CStdioFileEx(FILE\* pOpenStream);

CStdioFileEx(LPCTSTR lpszFileName, UINT nOpenFlags);

virtual ~CStdioFileEx();

virtual BOOL Open( LPCTSTR lpszFileName, UINT nOpenFlags, CFileException\* pError = NULL);

public:

//文件类型值转换到字符串

CString FileTypeToString();

//获取文件类型

TextCode GetFileType();

//按行读取文件

BOOL ReadString(CString& rString);

//静态方法 获取文件类型

static TextCode GetFileType( LPCTSTR lpszFileName);

protected:

TextCode m\_FileType;//保存文件类型

const static int PREDEFINEDSIZE;//预定义一行文件所需空间

protected:

//从UTF-8文件按行读取

BOOL ReadStringFromUTF8File(CString& rString);

//从ANSI文件按行读取

BOOL ReadStringFromAnsiFile(CString& rString);

//重UNCIDOE、UNICODE big endian文件读取

BOOL ReadStringFromUnicodeFile(CString& rString);

//UTF-8字符串转换到UNICODE字符串

CString UTF8ToUnicode(byte \*szUTF8);

//处理文件打开标志

UINT ProcessFlags(LPCTSTR lpszFileName, UINT& nOpenFlags,TextCode &tc);

};

#endif // !defined(AFX\_STDIOFILEEX\_H\_\_C1F1F96B\_9417\_4388\_8D24\_892EDFA2A616\_\_INCLUDED\_)

代码片段

// StdioFileEx.cpp: implementation of the CStdioFileEx class.

//

//////////////////////////////////////////////////////////////////////

#include "stdafx.h"

#include "StdioFileEx.h"

#ifdef \_DEBUG

#undef THIS\_FILE

static char THIS\_FILE[]=\_\_FILE\_\_;

#define new DEBUG\_NEW

#endif

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

/\*static\*/ const int CStdioFileEx::PREDEFINEDSIZE=1024;

CStdioFileEx::CStdioFileEx():CStdioFile()

{

m\_FileType=ANSI;//指定默认类型

}

CStdioFileEx::CStdioFileEx(FILE\* pOpenStream):CStdioFile(pOpenStream)

{

CString filepath=pOpenStream->\_tmpfname;//? 尚不清楚File\*结构

m\_FileType=GetFileType(filepath);

}

CStdioFileEx::CStdioFileEx(LPCTSTR lpszFileName, UINT nOpenFlags):CStdioFile(lpszFileName,ProcessFlags(lpszFileName, nOpenFlags,m\_FileType) )

{

}

CStdioFileEx::~CStdioFileEx()

{

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::GetFileType 静态方法 检测文本文件类型

// --------------------------------------------------------------------------------------------

/\*static \*/ TextCode CStdioFileEx::GetFileType(LPCTSTR lpszFileName)

{

CFile file;

byte buf[3];//unsigned char

TextCode tc;

try

{

if(file.Open(lpszFileName,CFile::modeRead|CFile::shareDenyNone|CFile::typeBinary))

{

file.Read(buf,3);

if(buf[0]==0xEF && buf[1]==0xBB && buf[2]==0xBF)

tc=UTF8;

else

if(buf[0]==0xFF && buf[1]==0xFE )

tc=UNICODE ;

else

if(buf[0]==0xFE && buf[1]==0xFF )

tc=UNICODEBIGENDIAN;

else

tc=ANSI;

}

else

tc=FILEERROR;

}

catch (CFileException ex)

{

CString errormsg;

errormsg.Format(\_T("操作文件%s时发生异常!"),ex.m\_strFileName);

AfxMessageBox(errormsg);

}

return tc;

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::Readstring 按行读取文本文件

//根据不同文件类型 调用不同的读取函数

// --------------------------------------------------------------------------------------------

BOOL CStdioFileEx::ReadString(CString& rString)

{

BOOL flag=FALSE;

switch(m\_FileType)

{

case ANSI:

flag=ReadStringFromAnsiFile(rString);

break;

case UNICODE:

case UNICODEBIGENDIAN:

flag=ReadStringFromUnicodeFile(rString);

break;

case UTF8:

flag=ReadStringFromUTF8File(rString);

break;

case FILEERROR:

flag=FALSE;

break;

default:

break;

}

return flag;

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::ReadstringFromAnsiFile 从ANSI文件读取字符串

// --------------------------------------------------------------------------------------------

BOOL CStdioFileEx::ReadStringFromAnsiFile(CString& rString)

{

BOOL flag;

try

{

flag=CStdioFile::ReadString(rString);

rString+="\r\n";

}

catch(CFileException ex)

{

CString errormsg;

errormsg.Format(\_T("操作文件%s时发生异常!"),ex.m\_strFileName);

AfxMessageBox(errormsg);

}

return flag;

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::ReadstringFromUTF8File 从UTF8文件中按行读取

//由于UTF-8编码多字节编码且各种字符长度不同,判断回车换行需要判断连续两个字节

// --------------------------------------------------------------------------------------------

BOOL CStdioFileEx::ReadStringFromUTF8File(CString& rString)

{

long index;

byte cr=0x0d;//回车换行符

byte lf=0x0a;

byte temp[2];

byte tempbyte;

byte \*pbuf=new byte[PREDEFINEDSIZE+1];

memset(pbuf,0,(PREDEFINEDSIZE+1)\*sizeof(byte));

UINT readlen;

try

{

//跳过文件头 移动文件指针

if (m\_pStream && ( GetPosition() == 0))

{

CStdioFile::Seek(3\*sizeof(byte),CFile::begin);

}

index=0;

do

{

memset(temp,0,2\*sizeof(byte));

readlen=CFile::Read(temp,2);//CStdioFile::Read效果不同 将省去回车符0x0d

if(!readlen)

return FALSE;

//元素存贮到字节数组中

pbuf[index++]=temp[0];

pbuf[index++]=temp[1];

tempbyte=temp[1];

//判断回车换行

if( ( tempbyte==cr && temp[0]==lf) ||(temp[0]==cr && temp[1]==lf))

break;

} while (readlen==2 && index<PREDEFINEDSIZE );

pbuf[index]=0;

rString=UTF8ToUnicode(pbuf);//UTF8编码转换到UNICODE

}

catch (CFileException ex)

{

CString errormsg;

errormsg.Format(\_T("操作文件%s时发生异常!"),ex.m\_strFileName);

AfxMessageBox(errormsg);

}

delete[] pbuf;

return TRUE;

}

// --------------------------------------------------------------------------------------------

//从UNICODE、UNICODE big endian文件按行读取

//当读取字节小于请求值(文件结束)或者超过预定义空间时无条件退出循环

//wChLine存放每行字符,wchtemp存放临时读取字符

//当编码为UNICODE big endian 时交换高低字节 ,将其转换成UNICODE字符串

// --------------------------------------------------------------------------------------------

BOOL CStdioFileEx::ReadStringFromUnicodeFile(CString& rString)

{

long index;

UINT readlen;

wchar\_t wchcr=MAKEWORD(0x0d,0x00);;//回车符 MakeWord(低、高字节顺序)

wchar\_t wchlf=MAKEWORD(0x0a,0x00);

wchar\_t \*wChLine=new wchar\_t[PREDEFINEDSIZE+1];

memset(wChLine,0,(PREDEFINEDSIZE+1)\*sizeof(wchar\_t));

wchar\_t wchtemp[2];

BOOL flag=TRUE;

try

{

//跳过文件头 移动文件指针

if (m\_pStream && ( GetPosition() ==0))

{

Seek(2\*sizeof(byte),CFile::begin);

}

index=0;

do

{

memset(wchtemp,0,2\*sizeof(wchar\_t));

readlen=CFile::Read(wchtemp,sizeof(wchar\_t)\*2);//CStdioFile::Read效果不同

if(!readlen)

break;

//UNICODE big endian交换高低字节

if(UNICODEBIGENDIAN==m\_FileType)

{

unsigned char high, low;

high = (wchtemp[0] & 0xFF00) >>8;

low = wchtemp[0] & 0x00FF;

wchtemp[0] = ( low <<8) | high;

high = (wchtemp[1] & 0xFF00) >>8;

low = wchtemp[1] & 0x00FF;

wchtemp[1] = ( low <<8) | high;

}

wChLine[index++]=wchtemp[0];

wChLine[index++]=wchtemp[1];

//判断回车换行

if(wchtemp[0]==wchcr && wchtemp[1]==wchlf)

break;

}

while( (readlen==sizeof(wchar\_t)\*2) && index<PREDEFINEDSIZE );

wChLine[index]=0;

CString strtext(wChLine,index);

rString=strtext;

if(rString.IsEmpty())

flag=FALSE;

}

catch (CFileException ex)

{

CString errormsg;

errormsg.Format(\_T("操作文件%s时发生异常!"),ex.m\_strFileName);

AfxMessageBox(errormsg);

}

delete[] wChLine;

return flag;

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::UTF8ToUnicode UTF-8字符串转换成UNICODE字符串

// --------------------------------------------------------------------------------------------

CString CStdioFileEx::UTF8ToUnicode(byte \*szUTF8)

{

CString strret;

strret=\_T("");

if(!szUTF8)

return strret;

//获取转换后所需串空间的长度

int wcsLen = MultiByteToWideChar(CP\_UTF8,0,(LPSTR)szUTF8,strlen((char\*)szUTF8),NULL,0);

LPWSTR lpw=new WCHAR[wcsLen+1];

if(!lpw)

return strret;

memset(lpw,0,(wcsLen+1)\*sizeof(wchar\_t));

//实施转换

MultiByteToWideChar(CP\_UTF8,0, (LPSTR)szUTF8,

strlen((char \*)szUTF8), (LPWSTR)lpw, wcsLen);

CString str(lpw);

delete[] lpw;

return str;

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::GetFileType获取文件类型

// --------------------------------------------------------------------------------------------

TextCode CStdioFileEx::GetFileType()

{

return m\_FileType;

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::FileTypeToString 文件类型枚举值转换为字符串值

// --------------------------------------------------------------------------------------------

CString CStdioFileEx::FileTypeToString()

{

CString strtype;

switch(m\_FileType)

{

case ANSI:

strtype.Format("%s",\_T("ANSI"));

break;

case UTF8:

strtype.Format("%s",\_T("UTF8"));

break;

case UNICODE:

strtype.Format("%s",\_T("UNICODE"));

break;

case UNICODEBIGENDIAN:

strtype.Format("%s",\_T("UNICODE big endian"));

break;

case FILEERROR:

strtype.Format("%s",\_T("FILEERROR"));

break;

default:

break;

}

return strtype;

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::Open 重载父类的文件打开操作 改变不同类型文件的打开方式

// --------------------------------------------------------------------------------------------

BOOL CStdioFileEx::Open( LPCTSTR lpszFileName, UINT nOpenFlags, CFileException\* pError)

{

ProcessFlags(lpszFileName,nOpenFlags,m\_FileType);//处理文件打开方式

return CStdioFile::Open(lpszFileName, nOpenFlags,pError);

}

// --------------------------------------------------------------------------------------------

//CStdioFileEx::ProcessFlags 处理不同文件的打开方式

//ANSI文件采用文本读取，UNICODE、UNICDOE big endian、UTF-8采用二进制方式读取

// --------------------------------------------------------------------------------------------

UINT CStdioFileEx::ProcessFlags(LPCTSTR lpszFileName, UINT& nOpenFlags,TextCode &tc)

{

tc=CStdioFileEx::GetFileType(lpszFileName);

if ((nOpenFlags & CFile::modeReadWrite)|| (nOpenFlags & CFile::modeRead) )

{

switch(tc)

{

case ANSI:

nOpenFlags|= CFile::typeText;

nOpenFlags&=~CFile::typeBinary;

break;

case UTF8:

nOpenFlags |= CFile::typeBinary;

nOpenFlags&= ~CFile::typeText;

break;

case UNICODE:

nOpenFlags |= CFile::typeBinary;

nOpenFlags&= ~CFile::typeText;

break;

case UNICODEBIGENDIAN:

nOpenFlags |= CFile::typeBinary;

nOpenFlags&= ~CFile::typeText;

break;

case FILEERROR:

break;

default:

break;

}

}

nOpenFlags|=CFile::shareDenyNone;

return nOpenFlags;

}

代码片段

void CReadStringDlg::OnBtnOpen()

{

// TODO: Add your control notification handler code here

char szFilter[] = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*||";

CFileDialog filedlg(TRUE , "txt", NULL, OFN\_HIDEREADONLY | OFN\_OVERWRITEPROMPT, szFilter,this);

if(IDOK==filedlg.DoModal())

{

m\_strPath=filedlg.GetPathName();

UpdateData(FALSE);

m\_ctrlEdit.SetSel(0,-1);

m\_ctrlEdit.Clear();

if(m\_stdiofileex.Open(m\_strPath,CFile::modeRead))

{

m\_strFileType=m\_stdiofileex.FileTypeToString();

UpdateData(FALSE);

}

else

{

MessageBox(\_T("读取文件失败!"));

}

}

}

//读取文件

void CReadStringDlg::OnBtnRead()

{

// TODO: Add your control notification handler code here

if(!ValidateInput())

return;

CString strread,strtemp;

m\_ctrlEdit.GetWindowText(strread);

m\_ctrlEdit.SetSel(0,-1);

m\_ctrlEdit.Clear();

if(m\_stdiofileex.m\_pStream)

{

int cnt=0;

strread+="\r\n";

while(cnt<m\_lLineCnt)

{

if(m\_stdiofileex.ReadString(strtemp))

{

strread+=strtemp;

cnt++;

}

else

{

AfxMessageBox(\_T("读取已经到达文件末尾!"));

break;

}

}

m\_ctrlEdit.SetSel(0,-1);

m\_ctrlEdit.ReplaceSel(strread);

}

else

{

MessageBox(\_T("读取文件失败!"));

}

}

//验证输入

BOOL CReadStringDlg::ValidateInput()

{

UpdateData();

if(m\_strPath.IsEmpty())

{

MessageBox("文件路径为空，请填写!");

return FALSE;

}

if(m\_lLineCnt<=0)

return FALSE;

return TRUE;

}

代码片段

PTSTR Normalise(PBYTE pBuffer

)

{

PTSTR ptText; // pointer to the text char\* or wchar\_t\* depending on UNICODE setting

PWSTR pwStr; // pointer to a wchar\_t buffer

int nLength; // a useful integer variable

// obtain a wide character pointer to check BOMs

pwStr = reinterpret\_cast<PWSTR>(pBuffer);

// check if the first word is a Unicode Byte Order Mark

if (\*pwStr == 0xFFFE || \*pwStr == 0xFEFF)

{

// Yes, this is Unicode data

if (\*pwStr++ == 0xFFFE)

{

// BOM says this is Big Endian so we need

// to swap bytes in each word of the text

while (\*pwStr)

{

// swap bytes in each word of the buffer

WCHAR wcTemp = \*pwStr >> 8;

wcTemp |= \*pwStr << 8;

\*pwStr = wcTemp;

++pwStr;

}

// point back to the start of the text

pwStr = reinterpret\_cast<PWSTR>(pBuffer + 2);

}

#if !defined(UNICODE)

// This is a non-Unicode project so we need

// to convert wide characters to multi-byte

// get calculated buffer size

nLength = WideCharToMultiByte(CP\_UTF8, 0, pwStr, -1, NULL, 0, NULL, NULL);

// obtain a new buffer for the converted characters

ptText = new TCHAR[nLength];

// convert to multi-byte characters

nLength = WideCharToMultiByte(CP\_UTF8, 0, pwStr, -1, ptText, nLength, NULL, NULL);

#else

nLength = wcslen(pwStr) + 1; // if Unicode, then copy the input text

ptText = new WCHAR[nLength]; // to a new output buffer

nLength \*= sizeof(WCHAR); // adjust to size in bytes

memcpy\_s(ptText, nLength, pwStr, nLength);

#endif

}

else

{

// The text data is UTF-8 or Ansi

#if defined(UNICODE)

// This is a Unicode project so we need to convert

// multi-byte or Ansi characters to Unicode.

// get calculated buffer size

nLength = MultiByteToWideChar(CP\_UTF8, 0, reinterpret\_cast<PCSTR>(pBuffer), -1, NULL, 0);

// obtain a new buffer for the converted characters

ptText = new TCHAR[nLength];

// convert to Unicode characters

nLength = MultiByteToWideChar(CP\_UTF8, 0, reinterpret\_cast<PCSTR>(pBuffer), -1, ptText, nLength);

#else

// This is a non-Unicode project so we just need

// to skip the UTF-8 BOM, if present

if (memcmp(pBuffer, "\xEF\xBB\xBF", 3) == 0)

{

// UTF-8

pBuffer += 3;

}

nLength = strlen(reinterpret\_cast<PSTR>(pBuffer)) + 1; // if UTF-8/ANSI, then copy the input text

ptText = new char[nLength]; // to a new output buffer

memcpy\_s(ptText, nLength, pBuffer, nLength);

#endif

}

// return pointer to the (possibly converted) text buffer.

return ptText;

}