**文本文件的读写**

**正确的文本文件读写过程**

1.定义文件变量；2.打开指定的文件；3.向从文本文件中写入信息；4.从文本文件中读取信息；5.关闭文件

**1、定义文件变量**

定义文件变量格式：CStdioFile 文件变量；

例如，定义一个名称为f1的文件变量，语句如下：CStdioFile f1;

**2、打开指定文件**

可以直接通过CStdioFile的构造函数来打开磁盘文件，同时可以用标志位指定打开方式（只读、只写、读写等）：

CStdioFile(LPCTSTR lpszFileName,UINT nOpenFlags);

其中，lpszFileName表示要打开的文件名，可以是相对路径或绝对路径

nOpenFlags设置文件打开方式标志位，可以指定用“|”连接多个标志位。下面是常用的打开标志：

CFile::typeText：以文本文件的形式打开文件

CFile::typeBinary：以二进制文件的形式打开文件

CFile::modeCreate：如果指定文件名的文件不存在，则新建文件；如果文件存在并且没有设置CFile::modeNoTruncate标志，则清空文件。

CFile::modeNoTruncate：如果文件存在，不把它的长度删除为0（即不清空文件中的数据）。

CFile::modeRead：以只读方式打开文件

CFile::modeReadWrite：以可读可写方式打开文件

CFile::modeWrite：以只写方式打开文件

CFile::shareDenyNone：文件打开后，不禁止其他进程对文件的读写操作

CFile::shareExclusive：文件打开后，禁止其他进程对文件的读写操作

CFile::shareDenyRead：文件打开后，禁止其他进程对文件的读操作

CFile::shareDenyWrite：文件打开后，禁止其他进程对文件的写操作

此外，可以不在构造函数中打开文件，而仅仅调用空的构造函数CStidoFile（），然后用CStdioFile::Open()打开文件。Open函数的前两个参数和非空构造函数的参数相同，其声明如下：

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

第3个参数与打开失败时的异常处理有关。

**实例1：以只读方式打开一个文件**

步骤：

使用AppWizard创建一个对话框应用程序，删除其自动产生的所有控件，添加一个Button控件。双击控件，在相应的函数里添加代码：

char \* pszFileName="C:\\myfile.txt";

CStdioFile myFile;

CFileException fileException;

if(!myFile.Open(pszFileName,CFile::modeCreate|CFile::typeText|CFile::modeRead),&fileException)

{

TRACE("Can't open file %s, error = %u\n",pszFileName,fileException.m\_cause);

}

运行结果：如果C:\下没有myfile.txt文件，则新生成该文件。

**3.向从文本文件中写入信息**

CStdioFile提供了函数WriteString来向文本文件中写入文本，WriteString函数的格式如下：

void WriteString(LPCTSTR lpsz);

WriteString的参数lpsz是一个以”\0”字符结束的字符串，要把这个字符串的内容写入文件。

**提示**：使用WriteString函数时，如果希望每执行一次WriteString，文本文件中的内容就会自动换行一次，那么就需要在需要换行的地方输出“\n”：

myFile.WriteString(“第1行\n”)；

**实例2：向文件中写入文本**

建立MFC基于对话框的程序，删除自动添加的所有控件，添加一个“确定”按钮，双击按钮，按默认添加事件函数，双击按钮，在相应的函数处添加如下代码：

char\* pszFileName="C:\\myfile.txt";

CStdioFile myFile;

CFileException fileException;

if(myFile.Open(pszFileName,CFile::typeText|CFile::modeCreate|CFile::modeReadWrite),&fileException)

{

myFile.WriteString("第1行\n");

CString strOrder;

strOrder.Format("%d,%.3f",66,88.88);

myFile.WriteString(strOrder);

}

else

{

TRACE("Can't open file %s,error=%u\n",pszFileName,fileException.m\_cause);

}

程序运行结果：C:\myfile.txt文件中内容如下：

第1行

66,88.880

**4.从文本文件中读取信息**

CStidoFile提供了函数ReadString来读取文本，ReadString有两种形式，一种为：

virtual LPTSTR ReadString(LPTSTR lpsz, UINIT nMax);

ReadString函数的参数如下：

lpsz ：是用户提供的一个指向字符串的指针，它用来接受从文件读出的文本，以”\0”结束。

nMax是本次所允许读入的文本字符个数，不计“\0”字符，也就是说最多能读入nMax-1个文本字符。

ReadString的返回值是一个LPTSTR类型的指针，它指向从文件读出的文本字符串，如果到达文件尾，则返回NULL。

ReadString的另一种形式为：

BOOL ReadString(CString& rString);

参数rString用来容纳从文件读出的文本。

CString版本忽略回车换行符，返回值是一个布尔值。如果返回值为FALSE，表示因到达文件尾而没有读到任何字符。

**提示：**每执行一次ReadString，就会自动从文本文件中读取一行数据，同时文件操作指针会自动跳转到下一行。

**实例3：从文件中读取文本信息**

步骤：创建基于对话框的MFC程序，删除所有自动添加的控件，添加按钮控件，为按钮添加事件，并在相应的函数处，添加如下代码：

char\* pszFileName="C:\\myfile.txt";

CStdioFile myFile;

CFileException fileException;

if(myFile.Open(pszFileName,CFile::typeText|CFile::modeReadWrite),&fileException)

{

myFile.SeekToBegin();

CString str1;

myFile.ReadString(str1);

CString str2;

myFile.ReadString(str2);

AfxMessageBox(str1+str2);

}

else

{

TRACE("Can't open file %s,error=%u\n",pszFileName,fileException.m\_cause);

}

myFile.Close();

**5.关闭文件**

对文件的操作完成后，使用CloseFile关闭文件。

函数CStdioFile::Close关闭一个文件，一般一个文件使用完毕就应该关闭它：

myFile.Close();

**错误的文本文件读写过程**

在读写文本文件的时候，最常见的错误是---操作文件不存在。这种错误产生的典型原因有：

**1.路径错误**

char \* pszFileName="C:\\Windows\\MyFile.txt";

CStdioFile myFile;

CFileException fileException;

if(!myFile.Open(pszFileName,CFile::modeCreate|CFile::typeText|CFile::modeReadWrite),&fileException)

{

//文件操作代码

}

else

{

TRACE("Can't open file %s, error = %u\n",pszFileName,fileException.m\_cause);

}

myFile.Close();

由于将文件变量与一个绝对路径的文件名关联，而程序的数据通常存储在相对路径下，所以一旦相对路径和相对路径不一致时，就会出错。

举例而言，上一段程序本意是想从windows的安装目录下面的MyTextFile.txt文件中读取一行数据，但是如果操作系统安装的路径不是C:\Windwos，而是C:\Winnt,那么这段程序就会出错。

解决方法是在程序中使用相对路径，改正后的程序如下：

//获取windows路径

LPTSTR lpBuffer=new char[MAX\_PATH];

::GetWindowsDirectory(lpBuffer,MAX\_PATH);

strcat(lpBuffer,"\\MyFile.txt");

CStdioFile myFile;

CFileException fileException;

if(myFile.Open(lpBuffer,CFile::typeText|CFile::modeCreate|CFile::modeReadWrite),&fileException)

{

//文件操作代码

}

else

{

TRACE("Can't open file %s, error = %u\n",pszFileName,fileException.m\_cause);

}

myFile.Close();

由于将文件变量与一个绝对路径的文件名关联，而程序的数据通常存储在相对路径下，所以一旦相对路径和相对路径不一致时，就会出错。

举例而言，上一段程序本意是想从windows的安装目录下面的MyTextFile.txt文件中读取一行数据，但是如果操作系统安装的路径不是C:\Windwos，而是C:\Winnt,那么这段程序就会出错。

解决方法是在程序中使用相对路径，改正后的程序如下：

//获取windows路径

LPTSTR lpBuffer=new char[MAX\_PATH];

::GetWindowsDirectory(lpBuffer,MAX\_PATH);

strcat(lpBuffer,"\\MyFile.txt");

CStdioFile myFile;

CFileException fileException;

if(myFile.Open(lpBuffer,CFile::typeText|CFile::modeCreate|CFile::modeReadWrite),&fileException)

{

//文件操作代码

}

else

{

TRACE("Can't open file %s, error = %u\n",pszFileName,fileException.m\_cause);

}

CString strFileTitle="MyFile.txt";

CStdioFile myFile;

CFileException fileException;

if(myFile.Open(strFileTitle,CFile::typeText|CFile::modeReadWrite),&fileException)

{

//文件操作代码

myFile.WriteString("测试！");

}

else

{

TRACE("Can't open file %s, error = %u\n",pszFileName,fileException.m\_cause);

}

myFile.Close();

**2.操作文件不存在**

如果应用程序所有路径下面不存在MyFile.txt文件，那么在WriteString写入信息时就会出错。

解决办法就是在程序中打开文件前要检查是否存在此文件。如下程序：

CString strFileTitle="MyFile.txt";

CFileFind finder;

if(finder.FindFile(strFileTitle))

{

CStdioFile myFile;

CFileException fileException;

if(myFile.Open(lpBuffer,CFile::typeText|CFile::modeCreate|CFile::modeReadWrite),&fileException)

{

//文件操作代码

}

else

{

TRACE("Can't open file %s, error = %u\n",pszFileName,fileException.m\_cause);

}

}

else

{

TRACE("Can't find file %s\n",strFileTitle);

}

myFile.Close();

**3.超越文件权限进行读写操作**

在打开文件的过程中，通过参数指定了文件的读写权限，如果读写的操作没有和相应的权限对应，就会出现错误。

下面的程序就是典型的忽略了文件操作权限的例子：

CString strFileTitle="MyFile.txt";

CStdioFile myFile;

CFileException fileException;

if(myFile.Open(strFileTitle,CFile::typeText|CFile::modeCreate|CFile::NoTruncate|CFile::modeRead),&fileException)

{

//文件操作代码

myFile.WriteString("测试!");

}

else

{

TRACE("Can't open file %s,error=%u\n",strFileTitle,fileException.m\_cause);

}

myFile.Close();

**支招儿：**

**1.准确定位文件的路径**

操作文件的过程中，经常需要将文本文件放在程序自身的目录中，但是如果仅仅在程序中使用不指定任何路径信息的相对路径，如：

myFile.Open("MyFile.txt",CFile::modeCreate|CFile::typeText|CFile::modeReadWrite);

那么就有可能出现不能正确定位的情况，准确定位文件位置的方法是获得可执行程序自身的绝对路径，如：

TCHAR FilePath[MAX\_PATH];

GetModuleFileName(NULL,FilePath,MAX\_PATH);

(\_tcstchr(FilePath,'\\'))[1]=0;

lstrcat(FilePath,\_T("MyFile.txt"));

CStdioFile myFile;

CFileException fileException;

if(myFile.Open(FilePath,CFile::modeCreate|CFile::typeText|CFile::modeReadWrite),&fileException)

{

//文件操作代码

}

else

{

TRACE("Can't open file %s,error=%u\n",FilePath,fileException.m\_cause);

}

myFile.Close();

**2.读文本文件指定的一行，并得到文本文件的总行数。**

读文本文件指定的一行，并得到文本文件的总行数

要统计文本文件的总行数，可以从头逐行读，直到文件尾，程序：

CStdioFile myFile;

CFileException fileException;

if(myFile.Open("MyFile.txt",CFile::modeCreate|CFile::modeNoTruncate|CFile::typeText|CFile::modeReadWrite),&fileException)

{

CString strContent;

int order=1;

while(myFile.ReadString(strContent))

{

if(2==order)

{

AfxMessageBox(strContent);

}

order=order+1;

}

}

else

{

TRACE("Can't open file");

}

myFile.Close();