# 1.定义宽字符串

|  |
| --- |
|  |

# 2关于MessageBox函数的第一个参数的赋值问题

## 如果给他赋值一个窗口句柄，那个句柄对于的窗口就是消息框的所有者，消息框始终覆盖在窗口上面，如果传递NULL，那么消息框的所有者的桌面。有时候会很不方便。建议传递窗口句柄，这样子只有你销毁了一个消息框，才会有第二个消息框产生。

# 3.下面我们来学习一下读取文件内容的API

## 1.还是我们的Project1项目，我们把原来的代码注释了

|  |
| --- |
|  |

## 2.然后我们在源文件所在的目录下面新建一个txt2文件

|  |
| --- |
|  |
|  |

## 3.我们添加下面的读取代码，在读取文件之前，先要创建一个文件句柄，调用CreateFile函数创建新文件或者打开一个现存的文件

|  |
| --- |
|  |

### 注意：CreateFile既可以创建文件，又可以打开文件.我们的程序运行的时候需要把文件拷贝到exe文件所在的目录否则找不到文件。打开文件后，需要判断一下是否打开成功，不成功程序就需要返回。

## 4.打开成功后我们来读取，先定义一个读取缓冲区的大小

|  |
| --- |
|  |

## 5.读取文件的代码如下

|  |
| --- |
|  |

### 效果：

|  |
| --- |
|  |

### 注意：文本文件的字符编码需要改为ANSI否则有很多乱码。

# app.c的代码如下

|  |
| --- |
| #include<stdio.h>  #include<Windows.h>  /\*\*  在命令行里面使用windows数据类型  \*/  #define BUF\_SIZE 4096  int main()  {  //读取文件  HANDLE hFile = CreateFile(TEXT("test1.txt"), GENERIC\_READ| GENERIC\_WRITE, FILE\_SHARE\_READ,  NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL,NULL);  //判断是否打开成功，如果失败提出消息框，并且在命令行显示具体错误信息  if (hFile == INVALID\_HANDLE\_VALUE)  {  MessageBox(NULL, TEXT("read file failed..."), TEXT("read error"), MB\_OK);  printf("file error:%x",GetLastError());  return -1;  }  //打开文件成功，就可以读取了，需要使用循环  DWORD nIn;  CHAR buffer[BUF\_SIZE];  memset(buffer, 0, BUF\_SIZE);  while (ReadFile(  hFile,  buffer,  BUF\_SIZE,  &nIn,  NULL  ) && nIn > 0)  {  printf("%s\n", buffer);  }  //读完文件记得要关闭句柄  CloseHandle(hFile);  system("pause");  return 0;  } |