**注意，用本文的方法修改后，不要使用这个设置语句**

**OSGeo.GDAL.Gdal.SetConfigOption("SHAPE\_ENCODING", "UTF-8");**

1. **原文中说路径也会有问题，经测试，似乎路径都是正常的**
2. **按原文进行修改后，变为正常的功能有：**

**Feature.GetFieldAsString(0)**

**Feature.GetFieldDefnRef(0).GetName()**

**Layer.GetName()**

**Feature.SetField(0,”测试”)**

**但还有部分功能无法正常使用，主要是一些将字段名作为参数传入的函数，只要传入的参数里含中文，那么就会出错，比如(画横线的是第二部分已经解决了的内容)**

**~~Layer.CreateField()~~**

**~~Feature.GetFieldAsString(“测试”)~~**

**~~Layer.FindFieldIndex(“矿山名”,1)~~**

**Layer.SetAttributeFilter(“矿山名 = ‘a1’”);**

**3.最终状态：**

**Feature.GetFieldAsString(string)**

**Feature.GetFieldDefnRef(int).GetName()**

**Layer.GetName()**

**Feature.SetField(int,string)**

**Layer.CreateField(FieldDefn,int)**

**Layer.FindFieldIndex(string,int)**

**这些函数已经修改完成，经测试正常，以下函数还有一些问题**

**Layer.SetAttributeFilter("矿山名 like '%测试%'") //字段名称有中文时无法运行，字段值有中文不影响，这个未找到原因**

**Feature.SetField(string,\*) //由于这类函数太多了，一个一个改太麻烦，所以没改**

**Feature.GetFeildAs\*\*(string) //没改理由同上，这两类建议先使用FindFieldIndex获取字段序号，再用序号读**

**以下为转载：**

# GDAL C#读取shp中文属性值乱码问题

GDAL的C#版本读取shp中，如果属性值中含有中文，读出来有可能是乱码的问题，根据SWIG生成的C#代码调试发现问题所在，在Ogr.cs文件中有这么一个函数，代码如下：

internal static string Utf8BytesToString(IntPtr pNativeData)

{

if (pNativeData == IntPtr.Zero)

return null;

int length = Marshal.PtrToStringAnsi(pNativeData).Length; //问题在这句

byte[] strbuf = new byte[length];

Marshal.Copy(pNativeData, strbuf, 0, length);

return System.Text.Encoding.UTF8.GetString(strbuf);

}

问题就出现在上面这句中，如果shp文件中dbf的编码是utf-8的时候，pNativeData实际上就是GDAL库读取到的一个const char\*的地址，这个时候，使用Marshal.PtrToStringAnsi函数返回的长度会变小，下面举个栗子：   
如果GDAL的C++库返回的const char\* pszValue = “中”，那么pNativeData的值实际上就是pszValue这个地址，且用UTF8编码的时候，pszValue指针对应的内存中的信息应该是”E4 B8 AD 00 XX XX”，后面的XX表示其他的东西，只有前四个有用。   
这时pNativeData的值就是上面这个”E4 B8 AD 00 XX XX”字符串的地址，调用下面这句，返回的值居然是2。明明一个3的字符串为啥长度是2，就因为这里长度变短，导致后面转string的时候丢失了信息，造成了乱码。

int length = Marshal.PtrToStringAnsi(pNativeData).Length; //length = 2

本来想试下Marshal.PtrToString其他的函数，结果发现Marshal.PtrToStringAuto和Marshal.PtrToStringUni返回的长度都是16，就更不对了，没办法，用了一个笨办法，写了个循环来查找长度。修改后的代码如下：

internal static string Utf8BytesToString(IntPtr pNativeData)

{

if (pNativeData == IntPtr.Zero)

return null;

int nMaxLength = Marshal.PtrToStringAuto(pNativeData).Length;

int length = 0;//循环查找字符串的长度

for(int i=0; i<nMaxLength; i++)

{

byte []strbuf1 = new byte[1];

Marshal.Copy(pNativeData+i, strbuf1, 0, 1);

if(strbuf1[0] == 0)

{

break;

}

length++;

}

byte[] strbuf = new byte[length];

Marshal.Copy(pNativeData, strbuf, 0, length);

return System.Text.Encoding.UTF8.GetString(strbuf);

}

同理，将Gdal.cs、Osr.cs这两个文件里面的这个函数也修改，重新编译即可解决这个问题。

最后，如果与shp中dbf的编码使用GBK之类的编码，请设置SHAPE\_ENCODING这个配置项，如果shp文件夹中有同名的cpg文件则可以不用设置，但是确保cpg文件中写的编码与dbf中的实际编码相对应

# GDAL C#中文路径，中文属性名称乱码问题

昨天写的博客，将C#读取shp中文属性值乱码的问题应该可以解决，博客地址为：<http://blog.csdn.net/liminlu0314/article/details/54096119>，然后又[**测试**](http://lib.csdn.net/base/softwaretest)发现中文路径，中文属性值如果有中文时，也会出现乱码，具体表现为偶数个汉字没有问题，奇数个汉字会出现乱码。   
继续调试C#的源码，发现问题还是在于将C++库中返回的结果进行编码转换的时候导致的，源码位于OgrPINVOKE.cs文件中。截取原始的代码如下：

protected class SWIGStringHelper {

public delegate string SWIGStringDelegate(string message);

static SWIGStringDelegate stringDelegate = new SWIGStringDelegate(CreateString);

[DllImport("ogr\_wrap", EntryPoint="SWIGRegisterStringCallback\_Ogr")]

public static extern void SWIGRegisterStringCallback\_Ogr(SWIGStringDelegate stringDelegate);

static string CreateString(string cstring) //这个函数出问题

{

return cstring;

}

static SWIGStringHelper() {

SWIGRegisterStringCallback\_Ogr(stringDelegate);

}

}

出问题的函数在于CreateString，这个函数是一个类似C的回掉函数，直接将C库中返回的const char\*直接转为C#中的string进行返回，这样的结果就是如果C库中的const char\*以UTF8编码的话，直接返回的就是乱码，并且可能导致截断。所以就表现为图层名乱码，属性字段名乱码等问题。   
解决方案与昨天处理的类似，将C库中的const char\*用UTF8编码之后再返回应该就没问题了，修改后的代码如下所示：

protected class SWIGStringHelper {

public delegate string SWIGStringDelegate(IntPtr message);//此处修改参数类型

static SWIGStringDelegate stringDelegate = new SWIGStringDelegate(CreateString);

[DllImport("ogr\_wrap", EntryPoint="SWIGRegisterStringCallback\_Ogr")]

public static extern void SWIGRegisterStringCallback\_Ogr(SWIGStringDelegate stringDelegate);

static string CreateString(IntPtr pNativeData)//此处函数内容需要修改

{

if (pNativeData == IntPtr.Zero)

return "";

//下面这一长串就是获取C字符串的长度，用Marshal库中的函数获取的都有问题，所以就用下面的循环来自己找了，如果有更好的方案请告知。

int nAnsiLength = Marshal.PtrToStringAnsi(pNativeData).Length;

int nUniLength = Marshal.PtrToStringUni(pNativeData).Length;

int nMaxLength = (nAnsiLength > nUniLength) ? nAnsiLength : nUniLength;

int length = 0;

for (int i = 0; i < nMaxLength; i++)

{

byte[] strbuf1 = new byte[1];

Marshal.Copy(pNativeData + i, strbuf1, 0, 1);

if (strbuf1[0] == 0)

{

break;

}

length++;

}

byte[] strbuf = new byte[length];

Marshal.Copy(pNativeData, strbuf, 0, length);

return System.Text.Encoding.UTF8.GetString(strbuf);

}

static SWIGStringHelper() {

SWIGRegisterStringCallback\_Ogr(stringDelegate);

}

}

**注：1.这里的For循环，我用的是i<nAnsiLength\*2，貌似效果会更好一些**

**2. 如果是.net 2.0编译器，Marshal.Copy(pNativeData + I,strbuf1,0,1)会出错，可以改成下边的代码**

**IntPtr newptr = new IntPtr(pNativeData.ToInt64()+i);**

**Marshal.Copy(newptr,strbuf1,0,1);**

修改完保存，同样的问题有4个文件，分别是OgrPINVOKE.cs、GdalPINVOKE.cs、OsrPINVOKE.cs和GdalConstPINVOKE.cs。 将这四个文件中SWIGStringHelper类中的内容都按照上面这样修改，然后重新生成dll，即可。

PS：上面的修改完之后，调试时没有问题，直接运行的时候最后一个汉字可能会丢失，问题很奇怪，不知道为啥。

编译好的库已经上传，下载地址为：<http://download.csdn.net/detail/liminlu0314/9730475>   
PS：经过测试，发现上面丢失最后一个汉字的问题已经解决，之前是debug的版本，用release的就没有这个问题。

# 第二部分：关于字段名称包含中文时，部分函数乱码的进一步修改

上述修改完成后，涉及中文字段名的以下函数还是有问题

Layer.CreateField()

Feature.GetFieldAsString(“测试”)

Layer.FindFieldIndex(“矿山名”,1)

Feature.SetField(string,\*)

Layer.SetAttributeFilter(“矿山名 = ‘a1’”);

猜测原因为：GDAL的C#组件通过SWIG调用C++的代码，在这个过程中，如果C#的代码将字符串直接作为参数传递过去，那么C#会自动检测系统当前的默认编码（例如Encoding.GB2312），并使用这个编码方式将字符串转成二进制数组，再传递给C++的DLL，但C++的DLL中，认为传递的字符串参数都是使用UTF8编码后的二进制数组，对其进行反编码后，就会出现乱码。

针对这个猜测的原因，使用如下的思路进行修改：就是不给C#进行自动检测编码并转码的机会，直接在程序中使用UTF8的编码方式将字符串转到二进制数组，直接将这个数组作为参数传给C++，看是否能绕开乱码的问题

以如下方法进行修改：

1. 使用如下函数，在C#中将字符串转为IntPtr指针

byte[] tmpBytes1 = System.Text.Encoding.UTF8.GetBytes(field\_name);

byte[] tmpBytes = new byte[tmpBytes1.Length + 1];

tmpBytes1.CopyTo(tmpBytes, 0);

tmpBytes[tmpBytes.Length - 1] = 0;

IntPtr tmpPtr = Marshal.AllocHGlobal(tmpBytes.Length);

Marshal.Copy(tmpBytes, 0, tmpPtr, tmpBytes.Length);

猜测：上面的代码中，不转到IntPtr，直接使用tmpBytes数组作为参数也可能可以，但这个没有测试

2. 修改相关函数的参数表，将原有的string类型参数换成IntPtr的参数，具体改法参考下面的样例

## 样例

## 1. Feature.GetFieldAsString(string)

涉及两个地方的改动，一个是Feature.cs的GetfieldAsString方法，一个是OgrPINVOKE.cs的Feature\_GetFieldAsString\_\_SWIG\_1方法

**修改前：**

Feature.cs：

public string GetFieldAsString(string field\_name) {

/\* %typemap(csout) (const char \*utf8\_path) \*/

IntPtr cPtr = OgrPINVOKE.Feature\_GetFieldAsString\_\_SWIG\_1(swigCPtr, field\_name);

string ret = Ogr.Utf8BytesToString(cPtr);

if (OgrPINVOKE.SWIGPendingException.Pending) throw OgrPINVOKE.SWIGPendingException.Retrieve();

return ret;

}

OgrPINVOKE.cs

[DllImport("ogr\_wrap", EntryPoint="CSharp\_Feature\_GetFieldAsString\_\_SWIG\_1")]

public static extern IntPtr Feature\_GetFieldAsString\_\_SWIG\_1(HandleRef jarg1, string jarg2);

**修改后：**

Feature.cs：

public string GetFieldAsString(string field\_name) {

/\* %typemap(csout) (const char \*utf8\_path) \*/

byte[] tmpBytes1 = System.Text.Encoding.UTF8.GetBytes(field\_name);

byte[] tmpBytes = new byte[tmpBytes1.Length + 1];

tmpBytes1.CopyTo(tmpBytes, 0);

tmpBytes[tmpBytes.Length - 1] = 0;

IntPtr tmpPtr = Marshal.AllocHGlobal(tmpBytes.Length);

Marshal.Copy(tmpBytes, 0, tmpPtr, tmpBytes.Length);

IntPtr cPtr = OgrPINVOKE.Feature\_GetFieldAsString\_\_SWIG\_1(swigCPtr, tmpPtr);

string ret = Ogr.Utf8BytesToString(cPtr);

if (OgrPINVOKE.SWIGPendingException.Pending) throw OgrPINVOKE.SWIGPendingException.Retrieve();

return ret;

}

OgrPINVOKE.cs

[DllImport("ogr\_wrap", EntryPoint="CSharp\_Feature\_GetFieldAsString\_\_SWIG\_1")]

public static extern IntPtr Feature\_GetFieldAsString\_\_SWIG\_1(HandleRef jarg1, IntPtr jarg2);

## 2. Layer.CreateField()

涉及两个地方的改动，FieldDefn.cs，第二个是OgrPINVOKE.cs

**修改前：**

FieldDefn.cs

public FieldDefn(string name\_null\_ok, FieldType field\_type) : this(OgrPINVOKE.new\_FieldDefn(name\_null\_ok, (int)field\_type), true, null) {

if (OgrPINVOKE.SWIGPendingException.Pending) throw OgrPINVOKE.SWIGPendingException.Retrieve();

}

public void SetName(string name) {

OgrPINVOKE.FieldDefn\_SetName(swigCPtr, name);

if (OgrPINVOKE.SWIGPendingException.Pending) throw OgrPINVOKE.SWIGPendingException.Retrieve();

}

OgrPINVOKE.cs

[DllImport("ogr\_wrap", EntryPoint="CSharp\_FieldDefn\_SetName")]

public static extern void FieldDefn\_SetName(HandleRef jarg1, string jarg2);

**修改后：**

FieldDefn.cs

public FieldDefn(string name\_null\_ok, FieldType field\_type) : this(OgrPINVOKE.new\_FieldDefn(name\_null\_ok, (int)field\_type), true, null) {

//为添加字段时，中文名称正常，这里重新设置一下正确的中文字段名

this.SetName(name\_null\_ok);

if (OgrPINVOKE.SWIGPendingException.Pending) throw OgrPINVOKE.SWIGPendingException.Retrieve();

}

public void SetName(string name) {

byte[] tmpBytes1 = System.Text.Encoding.UTF8.GetBytes(name);

byte[] tmpBytes = new byte[tmpBytes1.Length + 1];

tmpBytes1.CopyTo(tmpBytes, 0);

tmpBytes[tmpBytes.Length - 1] = 0;

IntPtr tmpPtr = Marshal.AllocHGlobal(tmpBytes.Length);

Marshal.Copy(tmpBytes, 0, tmpPtr, tmpBytes.Length);

OgrPINVOKE.FieldDefn\_SetName(swigCPtr, tmpPtr);

if (OgrPINVOKE.SWIGPendingException.Pending) throw OgrPINVOKE.SWIGPendingException.Retrieve();

}

OgrPINVOKE.cs

[DllImport("ogr\_wrap", EntryPoint="CSharp\_FieldDefn\_SetName")]

public static extern void FieldDefn\_SetName(HandleRef jarg1, IntPtr jarg2);

## 3. Layer.FindField(string,int)

涉及两个地方的改动，Layer.cs，第二个是OgrPINVOKE.cs

**修改前：**

Layer.cs

public int FindFieldIndex(string pszFieldName, int bExactMatch) {

int ret = OgrPINVOKE.Layer\_FindFieldIndex(swigCPtr, pszFieldName, bExactMatch);

if (OgrPINVOKE.SWIGPendingException.Pending) throw OgrPINVOKE.SWIGPendingException.Retrieve();

return ret;

}

OgrPINVOKE.cs

[DllImport("ogr\_wrap", EntryPoint="CSharp\_Layer\_FindFieldIndex")]

public static extern int Layer\_FindFieldIndex(HandleRef jarg1, string jarg2, int jarg3);

**修改后：**

Layer.cs

public int FindFieldIndex(string pszFieldName, int bExactMatch) {

byte[] tmpBytes1 = System.Text.Encoding.UTF8.GetBytes(pszFieldName);

byte[] tmpBytes = new byte[tmpBytes1.Length + 1];

tmpBytes1.CopyTo(tmpBytes, 0);

tmpBytes[tmpBytes.Length - 1] = 0;

IntPtr tmpPtr = Marshal.AllocHGlobal(tmpBytes.Length);

Marshal.Copy(tmpBytes, 0, tmpPtr, tmpBytes.Length);

int ret = OgrPINVOKE.Layer\_FindFieldIndex(swigCPtr, tmpPtr, bExactMatch);

if (OgrPINVOKE.SWIGPendingException.Pending) throw OgrPINVOKE.SWIGPendingException.Retrieve();

return ret;

}

OgrPINVOKE.cs

[DllImport("ogr\_wrap", EntryPoint="CSharp\_Layer\_FindFieldIndex")]

public static extern int Layer\_FindFieldIndex(HandleRef jarg1, IntPtr jarg2, int jarg3);

## 结果：

Feature.GetFieldAsString(string)

Feature.GetFieldDefnRef(int).GetName()

Layer.GetName()

Feature.SetField(int,string)

Layer.CreateField(FieldDefn,int)

Layer.FindFieldIndex(string,int)

这些函数已经修改完成，经测试正常，以下函数还有一些问题

Layer.SetAttributeFilter("矿山名 like '%测试%'") //字段名称有中文时无法运行，字段值有中文不影响，这个未找到原因

Feature.SetField(string,\*) //由于这类函数太多了，一个一个改太麻烦，所以没改

Feature.GetFeildAs\*\*(string) //没改理由同上，这两类建议先使用FindFieldIndex获取字段序号，再用序号读或写

# 第三部分：第二部分字段名称问题的另一种修改方法（经初步测试通过）

来源：

https://www.codeproject.com/Articles/138614/Advanced-Topics-in-PInvoke-String-Marshaling

大致原理为：

在将字符串作为参数传到C++代码之前，用Attribute指定一下Marshaler。这样基本原理与第二部分相同，但修改代码的工作量小一些。

流程：

## 在swig\csharp\ogr文件夹下新建一个UTF8Marshaler.cs

Namespace OSGEO.OGR{

using System;

using System.Runtime.InteropServices;

using System.Text;

public class UTF8Marshaler : ICustomMarshaler {

static UTF8Marshaler static\_instance;

public IntPtr MarshalManagedToNative(object managedObj) {

if (managedObj == null)

return IntPtr.Zero;

if (!(managedObj is string))

throw new MarshalDirectiveException(

"UTF8Marshaler must be used on a string.");

// not null terminated

byte[] strbuf = Encoding.UTF8.GetBytes((string)managedObj);

IntPtr buffer = Marshal.AllocHGlobal(strbuf.Length + 1);

Marshal.Copy(strbuf, 0, buffer, strbuf.Length);

// write the terminating null

Marshal.WriteByte(buffer + strbuf.Length, 0);

return buffer;

}

public unsafe object MarshalNativeToManaged(IntPtr pNativeData) {

byte\* walk = (byte\*)pNativeData;

// find the end of the string

while (\*walk != 0) {

walk++;

}

int length = (int)(walk - (byte\*)pNativeData);

// should not be null terminated

byte[] strbuf = new byte[length];

// skip the trailing null

Marshal.Copy((IntPtr)pNativeData, strbuf, 0, length);

string data = Encoding.UTF8.GetString(strbuf);

return data;

}

public void CleanUpNativeData(IntPtr pNativeData) {

Marshal.FreeHGlobal(pNativeData);

}

public void CleanUpManagedData(object managedObj) {

}

public int GetNativeDataSize() {

return -1;

}

public static ICustomMarshaler GetInstance(string cookie) {

if (static\_instance == null) {

return static\_instance = new UTF8Marshaler();

}

return static\_instance;

}

}

}

## 在需要进行转换的函数地方做一下声明

一般是在OgrPINVOKE.cs里调用dll中的C++函数，只需要修改这里即可，不需要修改Feature.cs或Layer.cs中的源代码，工作量会小一些

例如，原来的代码为：

[DllImport("ogr\_wrap", EntryPoint="CSharp\_Feature\_GetFieldAsInteger\_\_SWIG\_1")]

public static extern int Feature\_GetFieldAsInteger\_\_SWIG\_1(HandleRef jarg1, string jarg2);

可以看到第二个参数为string,修改为：

[DllImport("ogr\_wrap", EntryPoint="CSharp\_Feature\_GetFieldAsInteger\_\_SWIG\_1")]

public static extern int Feature\_GetFieldAsInteger\_\_SWIG\_1(HandleRef jarg1, [MarshalAs(UnmanagedType.CustomMarshaler,MarshalTypeRef=typeof(UTF8Marshaler))]string jarg2);

## 其它设置

要做一些其它设置，否则编译会报错，例如UTF8Marshaler.cs里用了unsafe代码，要添加一些参数

在swig\csharp\makefile.vc中，将

$(CSC) $(CSDEBUG) /target:library /out:ogr\_csharp.dll /r:osr\_csharp.dll ogr\\*.cs AssemblyInfo.cs

改成

$(CSC) $(CSDEBUG) /unsafe /target:library /out:ogr\_csharp.dll /r:osr\_csharp.dll ogr\\*.cs AssemblyInfo.cs