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**

**Layer.CreateField()**

**Feature.GetFieldAsString(“测试”)**

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

**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。