1.11版本编译流程与其它版本的流程大致相同，注意在C#编译时会报几个错误，解决方法如下，

接口重定义

osr\OsrPINVOKE.cs(192,10): error CS0111:

类型“OSGeo.OSR.OsrPINVOKE”已定义了一个名为“OsrPINVOKE”的具有相同参数类型的成员

osr\OsrPINVOKE.cs(188,10): (与前一个错误相关的符号位置)

NMAKE : fatal error U1077: “C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.EXE”: 返回代码“0x1”

解决办法比较简单，只需要进入到..\gdal-1.9.2\swig\csharp\gdal|ogr|osr三个文件夹下，找到 GdalPINVOKE.cs、OgrPINVOKE.cs、OsrPINVOKE.cs三个文件大约都是第188~192行，将下述重复的声明注释掉其中一个。

static GdalPINVOKE() {

}

//

static GdalPINVOKE() {

//}

三、

SWIG生成的接口成员名称错误，如：

gdal\Band.cs(17,79): error CS0117: “OSGeo.GDAL.GdalPINVOKE”并不包含“BandUpcast”的定义

gdal\Dataset.cs(17,82): error CS0117: “OSGeo.GDAL.GdalPINVOKE”并不包含“DatasetUpcast”的定义

gdal\Driver.cs(17,81): error CS0117:“OSGeo.GDAL.GdalPINVOKE”并不包含“DriverUpcast”的定义

NMAKE : fatal error U1077: “C:\Windows\Microsoft.NET\Framework\v4.

0.30319\csc.EXE”: 返回代码“0x1”

上述3个错误比较难定位，只能根据名称去相关类的文件里面搜索，经过反复查找发现SWIG-2.0.6/9均犯了同样的一个错误：在GdalPINVOKE.cs声明为下述三个名称的接口：

[DllImport("gdal\_wrap", EntryPoint="CSharp\_Driver\_SWIGUpcast")]

public static extern

IntPtr Driver\_SWIGUpcast(IntPtr jarg1);

[DllImport("gdal\_wrap", EntryPoint="CSharp\_Dataset\_SWIGUpcast")]

public static extern

IntPtr Dataset\_SWIGUpcast(IntPtr jarg1);

[DllImport("gdal\_wrap", EntryPoint="CSharp\_Band\_SWIGUpcast")]

public static extern IntPtr Band\_SWIGUpcast(IntPtr jarg1);

在Driver、Dataset、Band类中调用时竟然搞错了名字，写成了DriverUpcast、DatasetUpcast、BandUpcast，分别修改为下列名称即可。

public Driver(IntPtr cPtr, bool cMemoryOwn, object parent) : base(GdalPINVOKE.Driver\_SWIGUpcast(cPtr), cMemoryOwn, parent) {

swigCPtr = new HandleRef(this, cPtr);

public Dataset(IntPtr cPtr, bool cMemoryOwn, object parent) : base(GdalPINVOKE.Dataset\_SWIGUpcast(cPtr), cMemoryOwn, parent) {

swigCPtr = new HandleRef(this, cPtr);

public Band(IntPtr cPtr, bool cMemoryOwn, object parent) : base(GdalPINVOKE.Band\_SWIGUpcast(cPtr), cMemoryOwn, parent) {

swigCPtr = new HandleRef(this, cPtr);

1.11在中文路径方面已经做了修改，不修改代码的情况下，已经能够较好地支持中文路径，

但与之前不同的是，初始化时的代码需要变为这种：

OSGeo.GDAL.Gdal.SetConfigOption("GDAL\_FILENAME\_IS\_UTF8","YES");

与之前设置成NO有区别

此外，1.11版本的OGR库在读取和写入中文属性表时，会出现乱码的问题（字段名称读取写入无问题）

产生的原因为：GDAL在1.11后，在OGR库的C#代码OGR.cs中添加了StringToUtf8Bytes及Utf8BytesToString两个函数，读取和写入属性表内容时，会使用这两个函数对字符集编码进行转换，这个转换过程出了问题

由于这两个函数在其它地方也会用到（比如文件路径 转换等），所以不能直接修改这两个函数的内容，采用以下方式解决这个问题

1：在OGR.cs中添加两个函数，注意，除了字符集编码外，还需要注意字符长度（int length = Marshal.PtrToStringAuto(pNativeData).Length\*2;）的问题

private static System.Text.Encoding localEnc=null;

internal static byte[] StringToLocalBytes(string str)

{

if (str == null)

return null;

if(localEnc == null)

{

try

{

localEnc = System.Text.Encoding.GetEncoding("gb2312");

}

catch

{

localEnc = System.Text.Encoding.UTF8;

}

}

//int bytecount = System.Text.Encoding.UTF8.GetMaxByteCount(str.Length);

int bytecount = localEnc.GetMaxByteCount(str.Length);

byte[] bytes = new byte[bytecount + 1];

//System.Text.Encoding.UTF8.GetBytes(str, 0, str.Length, bytes, 0);

localEnc.GetBytes(str,0,str.Length,bytes,0);

return bytes;

}

internal static string LocalBytesToString(IntPtr pNativeData)

{

if (pNativeData == IntPtr.Zero)

return null;

if(localEnc == null)

{

try

{

localEnc = System.Text.Encoding.GetEncoding("gb2312");

}

catch

{

localEnc = System.Text.Encoding.UTF8;

}

}

int length = Marshal.PtrToStringAuto(pNativeData).Length\*2;

byte[] strbuf = new byte[length];

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

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

return localEnc.GetString(strbuf);

}

2. 在Feature.cs中的GetFieldAsString及SetField函数中，将原来的StringToUtf8Bytes及Utf8BytesToString调用改为上面添加的StringToLocalBytes及LocalBytesToString调用，如下(**改进：在GetFieldAsString里，可以在最后改成return ret.SubString(0,ret.IndexOf('\0'));** )：

public string GetFieldAsString(int id) {

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

IntPtr cPtr = OgrPINVOKE.Feature\_GetFieldAsString\_\_SWIG\_0(swigCPtr, id);

string ret = Ogr.LocalBytesToString(cPtr);

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

return ret;

}

public string GetFieldAsString(string name) {

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

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

string ret = Ogr.LocalBytesToString(cPtr);

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

return ret;

}

public void SetField(int id, string value) {

OgrPINVOKE.Feature\_SetField\_\_SWIG\_0(swigCPtr, id, Ogr.StringToLocalBytes(value));

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

}

public void SetField(string name, string value) {

OgrPINVOKE.Feature\_SetField\_\_SWIG\_1(swigCPtr, name, Ogr.StringToLocalBytes(value));

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

}

重新编译即可，两个CS文件的完整代码见附件