主要使用的方法原理参考另一篇笔记“C#开发奇技淫巧三：把dll放在不同的目录让你的程序更整洁”

一般来说，在C#中开发GDAL，主要用到的DLL文件包括以下几个



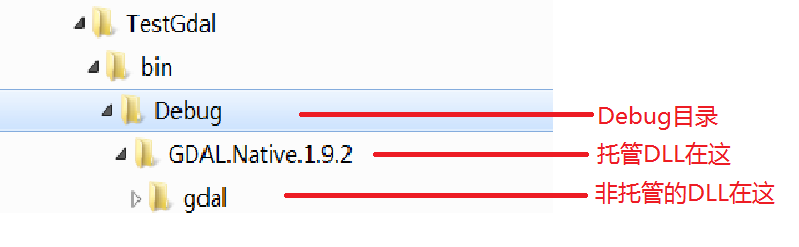
其中带csharp的为.net的托管DLL，其它均为非托管的DLL（可能还有一些其它的插件DLL，这些DLL都放到与gdal111.dll同目录下即可），如果需要将DLL文件从程序目录（例如Debug目录）移到其它位置（比如Debug中的某个子目录），需要做两点

1.将所有带csharp的托管DLL放到一个目录下，并使用上一篇笔记中的AssemblyResolve事件及委托函数，在程序中指定其路径

2.将其它所有非托管的DLL放到同一个目录下（这个目录可以与托管DLL相同，也可以不同），并使用更改PATH变量的方法，让程序在使用DllImport函数加载DLL时，能够找到这些DLL

详细如下

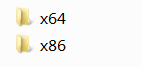
比如，我们将托管DLL放在Debug目录下的GDAL.Native.1.9.2子目录下，将非托管DLL放在GDAL.Native.1.9.2目录中的gdal子目录下，如下图



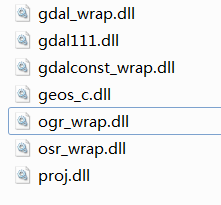
GDAL.Native.1.9.2文件夹下是这样



gdal文件夹下是这样



x64和x86文件夹下各放了一套对应平台的非托管DLL，



对于托管DLL，在程序加载时，使用AssemblyResolve事件及委托函数指定其位置，如下

[STAThread]

static void Main()

{

AppDomain.CurrentDomain.AssemblyResolve += new ResolveEventHandler(CurrentDomain\_AssemblyResolve);

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

static System.Reflection.Assembly CurrentDomain\_AssemblyResolve(object sender, ResolveEventArgs args)

{

System.Reflection.AssemblyName assemblyName = new System.Reflection.AssemblyName(args.Name);

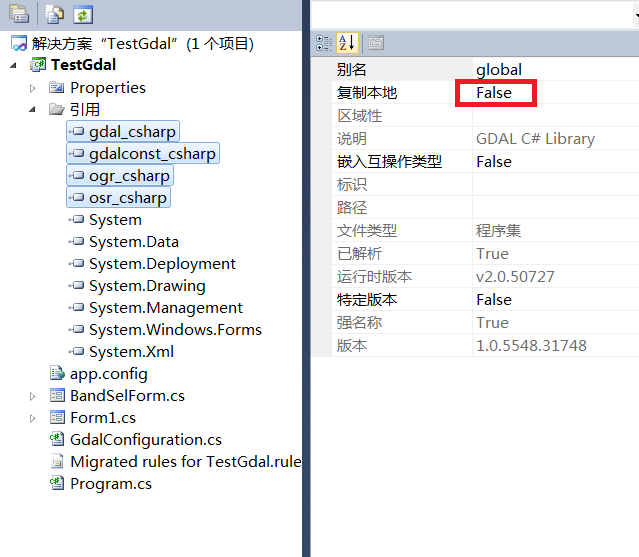
if (args.Name.Contains("gdal\_csharp") || args.Name.Contains("gdalconst\_csharp") || args.Name.Contains("ogr\_csharp") || args.Name.Contains("osr\_csharp"))

return System.Reflection.Assembly.LoadFrom(System.IO.Path.Combine(Application.StartupPath, "GDAL.Native.1.9.2", assemblyName.Name + ".dll"));

else return null;

}

同时，在解决方案管理器中，引用这四个dll时，将“复制本地”设置为false



对于非托管DLL，使用下面的代码配置一下PATH变量相关路径（该函数需要在调用GDAL库前进行调用，否则调用GDAL库将失败）

/// <summary>

/// Function to determine which platform we're on

/// </summary>

private static string GetPlatform()

{

return IntPtr.Size == 4 ? "x86" : "x64";

}

/// <summary>

/// Construction of Gdal/Ogr

/// </summary>

static GdalConfiguration()

{

var executingAssemblyFile = new Uri(Assembly.GetExecutingAssembly().GetName().CodeBase).LocalPath;

var executingDirectory = Path.GetDirectoryName(executingAssemblyFile);

if (string.IsNullOrEmpty(executingDirectory))

throw new InvalidOperationException("cannot get executing directory");

var gdalPath = Path.Combine(executingDirectory, @"GDAL.Native.1.9.2\gdal");

var nativePath = Path.Combine(gdalPath, GetPlatform());

// Prepend native path to environment path, to ensure the

// right libs are being used.

var path = Environment.GetEnvironmentVariable("PATH");

path = nativePath + ";" + Path.Combine(nativePath, "plugins") + ";" + path;

Environment.SetEnvironmentVariable("PATH", path);

// Set the additional GDAL environment variables.

var gdalData = Path.Combine(gdalPath, "data");

Environment.SetEnvironmentVariable("GDAL\_DATA", gdalData);

Gdal.SetConfigOption("GDAL\_DATA", gdalData);

var driverPath = Path.Combine(nativePath, "plugins");

Environment.SetEnvironmentVariable("GDAL\_DRIVER\_PATH", driverPath);

Gdal.SetConfigOption("GDAL\_DRIVER\_PATH", driverPath);

Environment.SetEnvironmentVariable("GEOTIFF\_CSV", gdalData);

Gdal.SetConfigOption("GEOTIFF\_CSV", gdalData);

var projSharePath = Path.Combine(gdalPath, "share");

Environment.SetEnvironmentVariable("PROJ\_LIB", projSharePath);

Gdal.SetConfigOption("PROJ\_LIB", projSharePath);

}