经测试，GDAL编译为64位时，暂时无法进行调试，编译为32位的时候，可以进行调试，

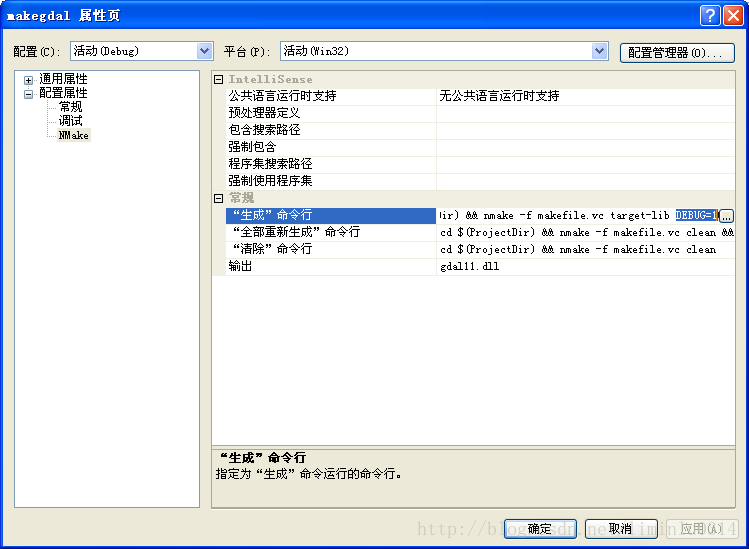
参考：

[GDAL库调试（包括跨语言调试）](http://blog.csdn.net/liminlu0314/article/details/9427263)

很多时候都需要调试GDAL库，尤其是像学习GDAL库中的某些**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "_blank)**是如何实现的时候，调试就必不可少了。

首先说明用C++的调试。以VS2008为例进行说明。

编译DEBUG版本的GDAL库，这个可以参考我之前的博客。可以用命令行编译，也可以用IDE编译，但是实质都是一样，修改 nmake命令的参数。也就是给nmake命令后加一句DEBUG=1即可。



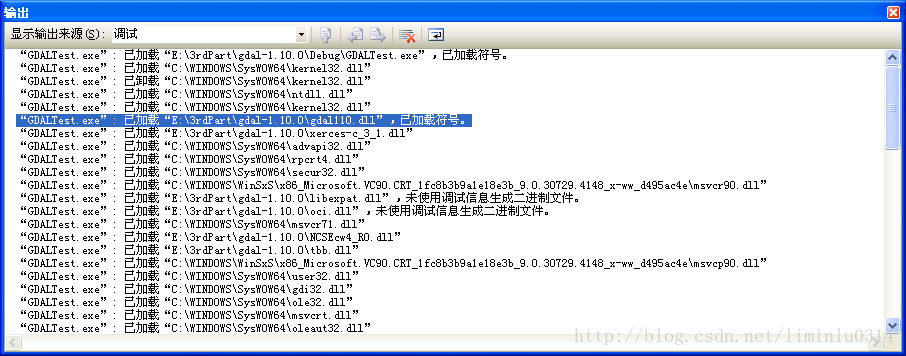
编译完成之后，应该会在gdal的源码目录中生成下面几个文件，如下图。



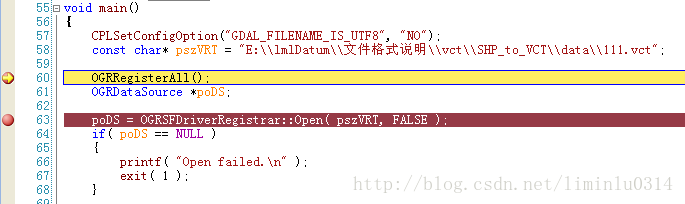
调试时用到的文件主要就是pdb文件，请确保pdb文件存在，大小应该在30Ｍ以上说明是可以调试的，有时候生成的pdb文件会小于10M，这时应该清理GDAL的工程，然后重新编译就好了。

将上面gdal110.dll、gdal110.pdb、gdal110.ilk等文件拷贝至你的测试工程中的exe所在目录，然后使用调试方式启动程序，在程序调用GDAL库中的函数时，按F11键应该就可以进入GDAL的源码文件中。

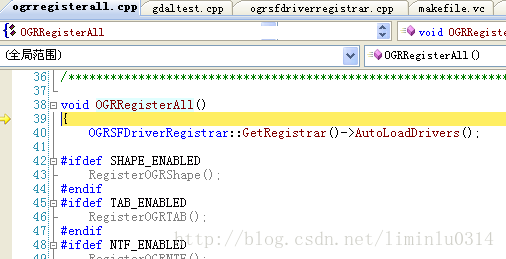
如果GDAL的pdb文件在程序启动时没有加载，就不能进入GDAL的源码。判断是否加载GDAL的pdb文件，可以在VS的输出窗口中看到“gdal110.dll已加载符号”的字样就说明加载成功，可以调试，如下图所示。



然后在程序中添加断点即可。在OGRRegisterAll函数处设置断点，如下图。



程序执行到该句之后，按F11键，程序自动会跳转到GDAL源码文件中，如下图所示：

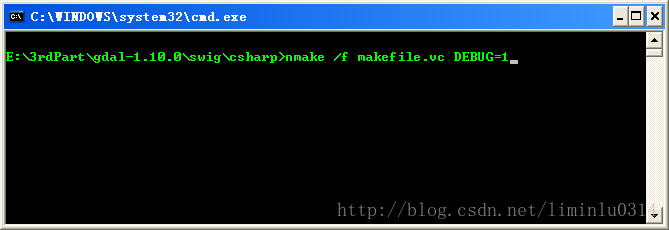


然后就可以调试GDAL的源代码了，其他函数类似。或者你提前将要调试的GDAL源码用VS打开，添加断点，然后直接按F5，不出意外的话，程序会在GDAL源码中设置的断点处进行暂停。

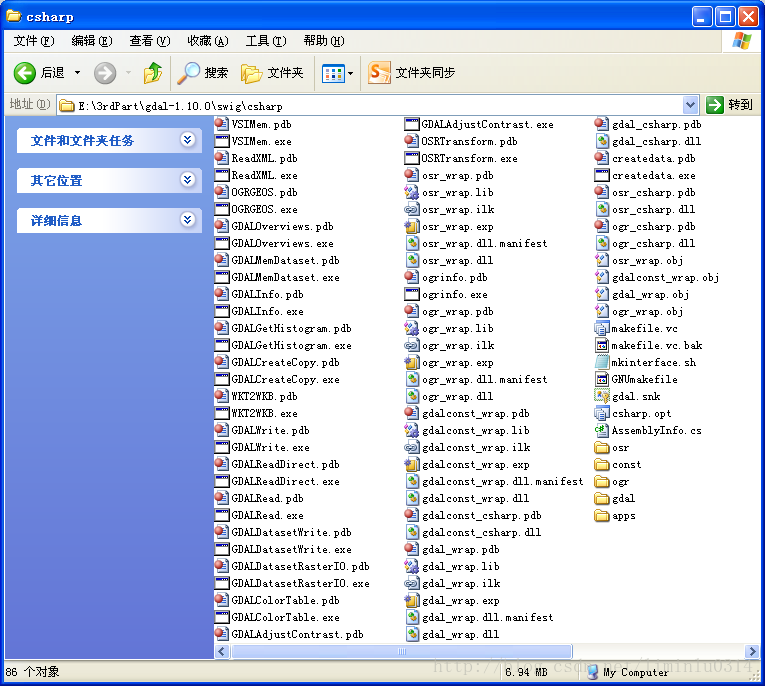
**跨语言调试（以C#为例）**

GDAL同时可以跨语言调试，下面以C#为例进行说明。

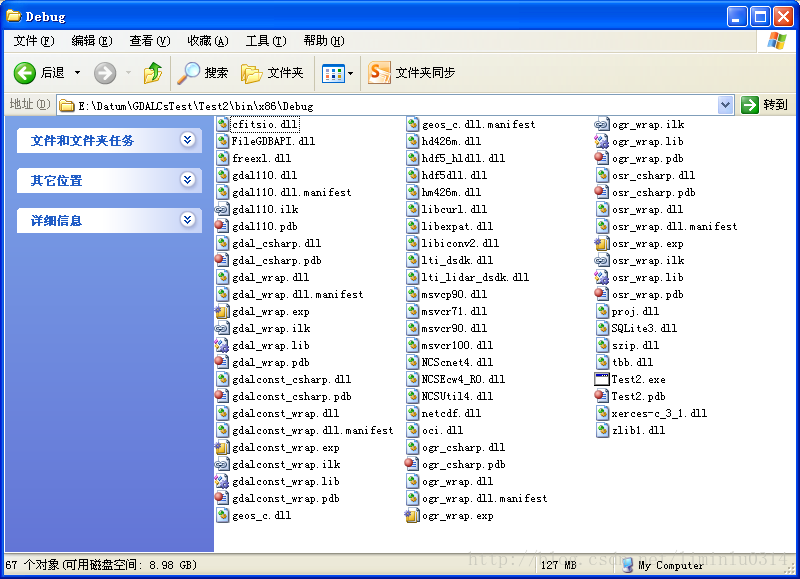
首先请确保编译的GDAL版本是debug的版本，具体方式参考上面或者之前的博客。编译完C++版本的Debug的GDAL库后，接下来编译C#版本，同样也要编译Debug版本。具体方式与编译C++类似，也是在编译时指定DEBUG=1即可。编译命令如下图：



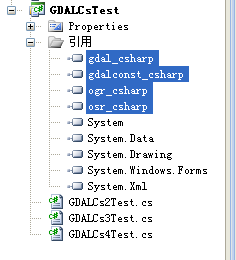
编译完之后，在gdal\swig\csharp目录中可以看到一大批exe、pdb文件。如果编译的不是Debug版本，是不会生成pdb文件的。可以通过是否生成pdb文件来判断编译的是否debug版本。如下图所示。



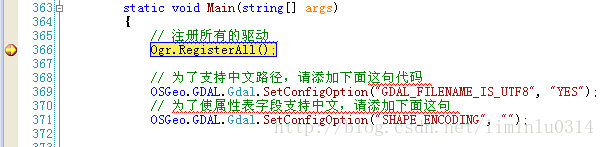
将上面编译的C#版本中的八个dll以及对应的pdb、lib、ilk、exp等文件一同拷贝到C#的测试程序中exe所在目录。如下图所示。



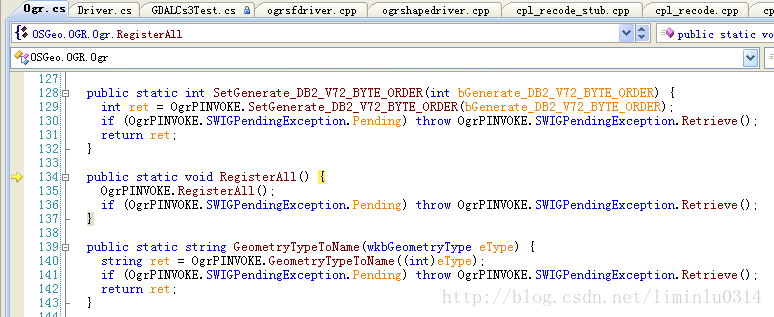
接下来在C#引用中，添加C#版本八个dll中文件名中带有\_csharp的四个dll。



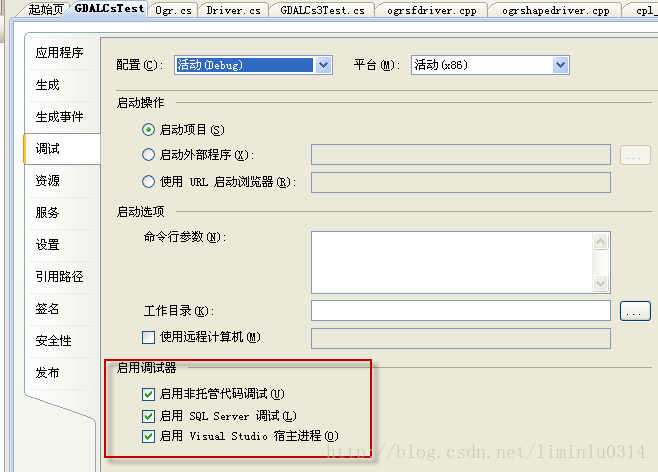
然后在C#代码中的Ogr.RegisterAll()函数处设置断点，如下图所示。



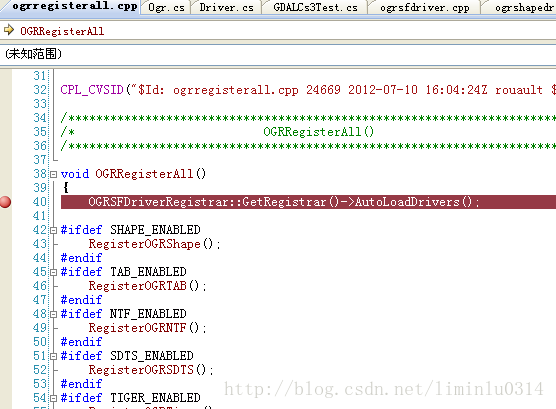
编译通过后，按F5启动调试模式。程序在断点处中断，然后按F11，VS会自动进入swig封装的C#代码中，如下图所示。



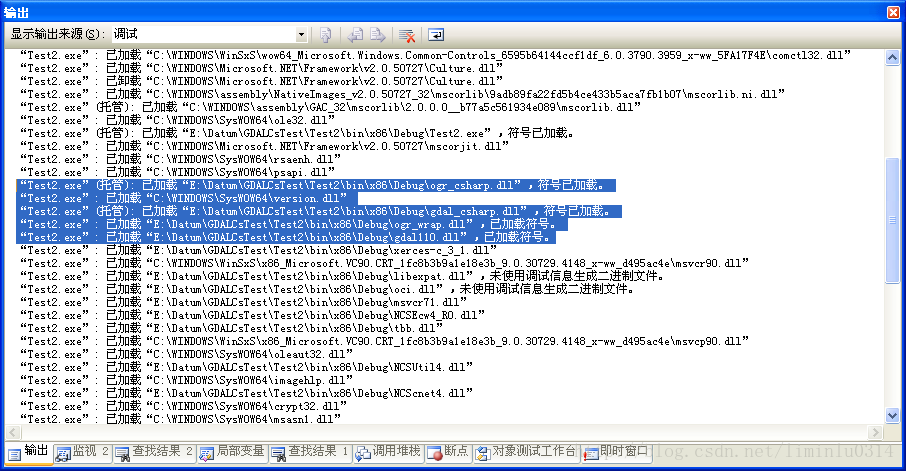
此时接下来，再按F11是不可能进入C++的源代码中。要进入C++的源代码中，要进行下面的设置。第一，设置C#的程序属性，启用非托管代码调试，如下图红色框区域所示。

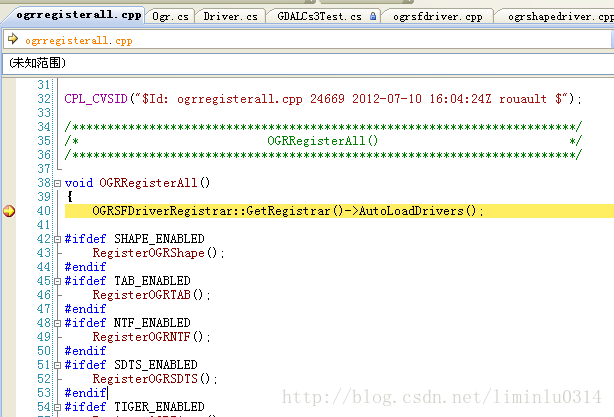


第二，需要提前将C++的源代码打开，比如函数RegisterAll的C++代码在文件gdal-1.10.0\ogr\ogrsf\_frmts\generic\ogrregisterall.cpp中，打开该文件，并在函数开头添加断点，如下图所示。



通过上面两步设置，就可以通过C#来调试GDAL的源代码了。与上面一样，按F5启动，同样可以通过输出窗口看到是否已经加载pdb文件了。

 启动后，直接按F5，程序就自动在C++的代码中设置的断点处进行中断，如下图所示。



来源： <http://blog.csdn.net/liminlu0314/article/details/9427263>