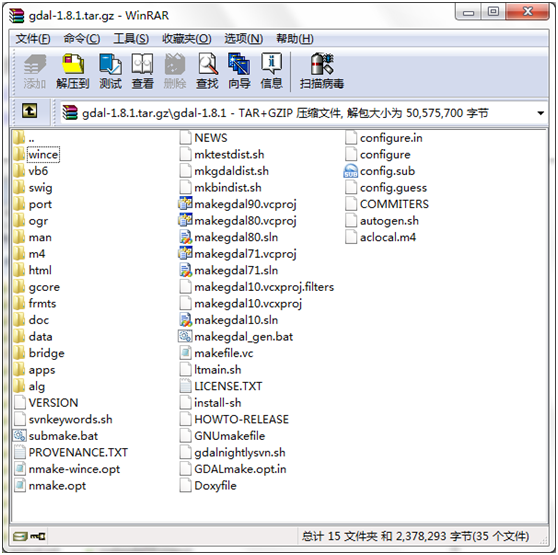
[GDAL源码剖析（一）](http://blog.csdn.net/liminlu0314/article/details/6933772)

资料来自http://blog.csdn.net/liminlu0314/article/details/6933772

一、GDAL简介

    GDAL官方网站<http://www.gdal.org/>，本文章中的基本内容都是参照官网中的信息，如有错误或者与官网中的内容冲突，以官网中的为正确。

在开始文章之前，我想先提出几个问题，什么是GDAL？GDAL能做什么？GDAL怎么使用？GDAL内部结构是怎么组织的？GDAL提供的算法原理是什么？对于上面的几个问题，希望大家看完该系列文章后能对上面的几个问题少点疑惑，希望能对感兴趣的童鞋们有所帮助。本人不才，文章中难免会出现问题，希望大家不吝指正。

什么是GDAL？这个问题比较简单，通俗的讲，GDAL是一个读写空间数据（这里的空间数据包括栅格数据和矢量数据）的开源库（但不仅限于此，此外还提供了一些非常常用的算法和工具）。严格的讲，大家可以参考GDAL首页上的介绍。

1、alg文件夹：alg文件夹中存放的是GDAL库中提供的一些算法的源代码，这些算法包括但不限于：DEM生成等高线算法；图像纠正算法（几何纠正，TPS纠正，正射RPC纠正）；栅格矢量化算法；矢量栅格化算法；格网计算算法；PCT和RGB互转算法；分类图的小碎斑块去除算法等。

2、apps文件夹：apps文件夹中存放的是GDAL库中提供的一些命令行工具集的源代码，这些工具集的介绍可以参考<http://gdal.org/gdal_utilities.html>，将来我会对这些工具做一个简单的说明。其中有些工具非常的有用，比如gdalinfo，可以使用该工具来查看图像的元数据信息等。

3、bridge文件夹：bridge文件夹中存放的是用来连接GDAL抽象类的定义以及GDAL自己的结构体定义和实现的源代码。具体到后面涉及到GDAL的实现原理时会对该文件夹作一个比较详细的介绍。

4、data文件夹：data文件夹中存放的是GDAL库中需要用到的一些“配置文件”（此处用配置文件可能不太准确），这些文件主要有ESRI的投影文件，ESPG的投影文件，PCI的投影和椭球体文件，autoCAD的头文件，以及其他的一些文件。在GDAL库中有很多时候会自动读取该文件夹中的文件，一般是通过环境变量来查找该文件夹，环境变量的名字叫GDAL\_DATA，值就是data文件夹的路径，或者可以在你的程序中使用函数 CPLSetConfigOption("GDAL\_DATA","C:\GDAL\data");来进行设置该文件夹的目录，如果没有设置GDAL会自动从环境变量中查找，如果还是没有找到，那么GDAL可能会提示错误，比如如果不设置GDAL\_DATA，那么在写如atuocad的dxf格式的时候就提示创建不成功，后面遇到的话会再进行说明。

5、doc文件夹：doc文件夹存放的是用来生产GDAL帮助文档的一些dox文件，dox文件是使用doxygen工具来进行生成的，后面会对doxygen工具作一个简单的介绍和说明，以及在自己的工程中怎么使用doxygen生成自己的程序的开发帮助文档等。总之一句话，这个文件夹就是用来生成GDAL库的帮助文档的一些东西。后面会告诉大家如何生成一份GDAL的帮助文档，当然你也可以把gdal.org整个网站抓下来，J

6、frmts文件夹：这个文件夹可以说是GDAL代码中东西最多的一个文件夹了，每次更新GDAL的版本后都会发现这个文件夹中会多出几个文件夹，同时GDAL也会中支持的文件格式中多出来几个新的文件格式。没错，这个文件夹存放的就是GDAL针对每种不同的特定的图像格式解析的源代码，可以举几个简单的例子，比如bmp文件夹就是解析BMP图像的，hfa文件夹就是用来解析Erdas的img图像格式，raw文件夹用来读取ENVI的hdr文件，还有pcidsk文件夹就是读取PCI的pix格式的等等。所以这个文件夹存放的是解析各个文件格式的源代码。

7、gcore文件夹：通过名字大家也应该知道这个文件夹是做什么的了，叫core的肯定都是很核心的东西了，这个文件夹就是GDAL的灵魂所在，主要存放的GDAL抽象类的数据集，波段，图像读写接口等都在这个里面实现的。如果要知道GDAL的抽象类是怎么对图像格式进行抽象的，可以看看这个里面的代码。

8、html文件夹：html文件夹如果使用压缩包的话，里面应该是空的，这个文件夹主要是用来存放GDAL的生成的帮助文档的地方，主要是使用前面介绍的doc文件夹中的dox脚本，使用doxygen工具生成的GDAL帮助文档会出现在这个文件夹中。后面会和doc文件夹一起进行一个详细介绍。

9、m4文件夹：m4文件夹存放的有好几个已m4为后缀名的文件，m4文件叫MacroProcessor Library，m4文件是编译基础中最核心的文件，这个文件主要是用autoconf来产生configure配置文件，继而自动生成Makefile文件。这个文件夹中Windows平台下貌似没什么作用，可能是我还不知道吧，在此略过。

10、man文件夹：man文件夹貌似是用来生成linux或者其他平台下的帮助文件，估计是可以使用linux下的man帮助吧。Windows平台下貌似也没什么用，略过。

11、ogr文件夹：用过GDAL的肯定知道ogr库吧，在很久很久以前，GDAL和OGR是两个库，GDAL专门负责读取栅格数据，OGR库负责读取矢量数据，然后可能是因为两个库分开有些不方便，比如GDAL的算法库中经常会用到矢量数据的读取，或者还有别的原因吧，现在将这两个库整合在了一起，目前OGR库就是GDAL库的一个子集。其实OGR库还是可以单独编译出来的。Ogr文件夹就是存放OGR库源代码的文件夹。这个文件夹里面也是有很多东西的，后面再详细进行介绍。

12、port文件夹：port文件夹中存放的是port库的东西，port库对于GDAL库来说是一个底层的支持库，port库中定义了一些字符串的操作，文件处理，网页请求，数据库连接，哈希表，字符加密文件压缩等基础的函数。比如GDAL中所有的导出函数符号CPL\_DLL就是在这个port文件夹中定义的，还有frmts文件夹中，打开文件，打开数据库，打开网络路径等都是用的port库，以及字符串的处理等。

13、swig文件夹：swig文件夹主要是存放swig的脚本。Swig全称叫SimplifiedWrapper and Interface Generator，网址是[www.swig.org](file:///F:\Work\Data\www.swig.org)，          swig的作用就是可以将C/C++写的库封装为Python，C#，Java，Perl和 Ruby等其他语言的访问接口。网上GDAL的C#版本就是使用swig来编译出来实现的，很强大吧。后面有时间的话，会写一篇关于swig编译GDAL的文章。

14、vb6文件夹：这个文件夹中用来将GDAL编译成一个VB6的模块，对于里面具体的文件说明，以及如何编译参考文件夹中的readme.txt，由于我对VB6的使用，还是仅限于拖个按钮，写个单击事件的基础，对于模块之间的调用，实在是不太懂，所在就不说明了。想用VB6使用GDAL的可以自己研究一下，应该也不是很难。

15、wince文件夹：顾名思义，这个文件夹中的东西就是用来编译Windows CE平台下的GDAL库用的，具体编译请参考其中的说明文档。

文件夹介绍完了，下面对文件进行一个大概说明：

1、aclocal.m4：同上面的m4文件夹

2、autogen.sh：Linux平台下的shell文件，用来调用autoconf来产生configure配置文件的。

3、COMMITERS：该文件中的内容是GDAL开发人员的信息，姓名，联系邮箱以及各自负责开发的模块说明等。

4、config.guess，config.sub，configure，configure.in：这四个文件貌似都是linux平台下的配置文件，中Windows下没啥用，略过。

5、Doxyfile：Doxyfile就是前面doc文件夹中说明提到的doxygen的工程文件，用来生成帮助文档用的，后面在介绍doxygen的使用是会对该文件进行一个说明。

6、GDALmake.opt.in：这个文件是linux平台下的GDAL库编译配置文件，功能在后面的nmake.opt中介绍。

7、gdalnightlysvn.sh：Linux平台下调用svn获取GDAL源代码的一个shell脚本。

8、GNUmakefile：GNU的make文件。

9、HOWTO-RELEASE：GDAL发布版本的一些说明。

10、    install-sh：GDAL的安装shell脚本，Linux平台下。

11、    LICENSE.TXT：GDAL的许可说明文件。

12、    ltmain.sh：libtool的shell脚本，Linux平台下，Windows下貌似没用到。

13、    makefile.vc：GDAL的编译文件，用来将源代码编译成dll文件，后面会在GDAL编译中作一个介绍。

14、    makegdal\_gen.bat：用来生成VS的工程文件的一个批处理文件，后面在GDAL编译中会对该文件的使用方式做一个说明。

15、    makegdal10.sln：文件夹中所有的sln文件都是VS的项目文件，文件名后的数字代表的是VS的版本号。

16、    makegdal10.vcxproj和makegdal71.vcproj：VS的工程文件，该文件可以由makegdal\_gen.bat文件自动生成，后面详细介绍。

17、    mkbindist.sh，mkgdaldist.sh和mktestdist.sh：三个shell脚本文件，Windows下没用，略过。

18、    NEWS：GDAL的新增功能，以及修复的bug记录等。

19、    nmake.opt：GDAL编译选项配置文件，在编译GDAL中，可以指定GDAL绑定的其他库等都在这个里面进行设置。在后面的GDAL编译中会详细介绍说明。

20、    nmake-wince.opt：编译wince版本的编译选项配置文件。

21、    PROVENANCE.TXT：GDAL目录说明文件，如果上面说明的不够清楚，可以参考这个文件。

22、    submake.bat：一个编译的批处理文件，目前没啥用。

23、    svnkeywords.sh：又是svn的一个shell脚本。

24、    VERSION：GDAL版本信息。

[GDAL源码剖析（二）之编译说明](http://blog.csdn.net/liminlu0314/article/details/6937194)

http://blog.csdn.net/liminlu0314/article/details/6937194

**一、简单的编译**

**1、使用VisualStudio IDE编译**

首先进入GDAL的源代码目录，可以看到有几个sln为后缀的文件名，比如makegdal10.sln，makegdal80.sln，makegdal71.sln，makegdal90.sln 。这些文件是VisualStudio的工程文件，后面的数字对应的VS的版本号，71表示的VS2003，80表示VS2005，90表示VS2008，还有10表示VS2010等。根据自己电脑安装的VS版本，打开对应的文件，如下图所示（使用VS2008SP1版本，打开makegdal90.sln文件）：

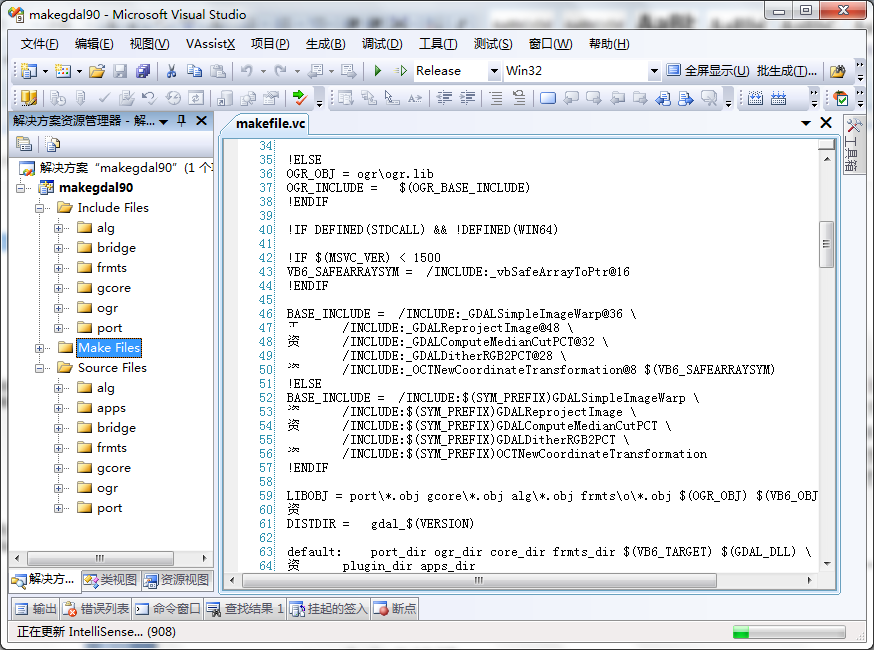
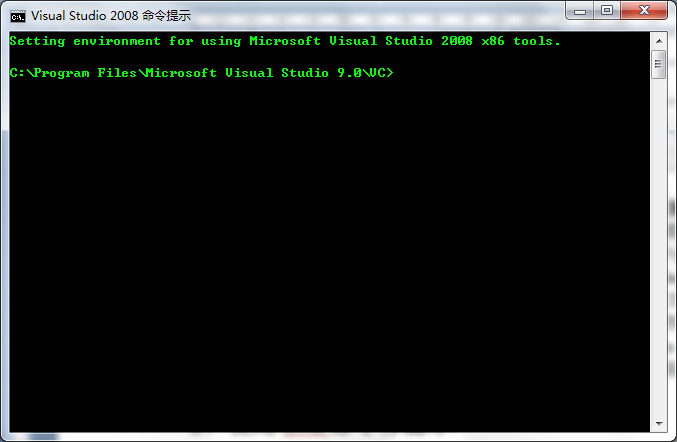


图3 VS2008打开编译GDAL1.8.1

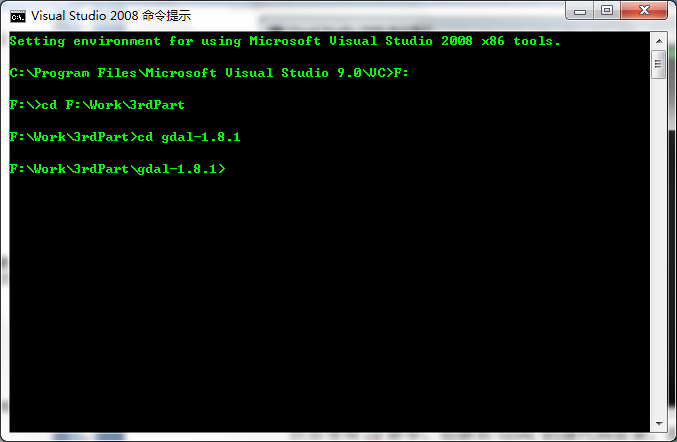
然后在左侧解决方案右键，弹出菜单中选择“生成”或者“重新生成”命令，然后GDAL就会开始编译，等待输出窗口中提示，执行完成，生成成功等信息后，就表示GDAL已经完成编译。同时会在GDAL的源代码目录中会出现gdal.lib，gdal\_i.lib，gdal18.dll等文件，如果你没有修改GDAL中的nmake.opt文件的话，那么同时会在你的C盘中会出现一个叫“C:\warmerda\bld”的文件夹，里面会包含三个文件夹，分别是bin，data和html。其中bin文件夹中存放的是编译出来的GDAL的可执行程序，包括GDAL提供的十几个工具集；data文件夹就是在第一节中的介绍的data文件夹；html文件夹中存放的是各种数据格式的说明文档。

## 2、使用cmd命令行编译

使用cmd命令行编译，首先在“开始菜单\所有程序\Microsoft Visual Studio 2008\Visual Studio Tools\ Visual Studio 2008命令提示”，点击“Visual Studio 2008 命令提示”会弹出下面的界面：



然后使用cd命令，切换到GDAL的源代码目录，如下图所示：



切换到GDAL的源代码目录后，依次敲入下面的命令行后回车，等待编译结束即可。

nmake -f makefile.vc

nmake -f makefile.vc install

nmake -f makefile.vc devinstall

同时还有其他的命令，如：

nmake -f makefile.vc clean

nmake -f makefile.vc MSVC\_VER=1400clean

nmake -f makefile.vc MSVC\_VER=1400DEBUG=1

       上面六行的命令含义依次是：

编译GDAL库

编译GDAL库，并安装（这里安装的意思就是将生成的dll，exe等文件拷贝到C:\warmerda\bld目录），

编译GDAL库，并安装开发者模式（安装的意思同上，开发者模式意思是将开发用的include文件夹中的头文件和lib文件一同拷贝到C:\warmerda\bld目录，此时会在C:\warmerda\bld目录中多出来两个文件夹，分别是include和lib，分别存放的是GDAL的头文件和lib文件，用于调用GDAL库使用）。

清理GDAL库，同时会删除编译GDAL库所生成的临时文件，作用相当于在VS环境中的清理命令。

作用同上，但是添加了一个MSVC\_VER=1400，表示使用VS2005编译。

编译GDAL库的debug模式，可以用来调试GDAL源码。

# 二、自定义编译

GDAL的强大之处不单单在于可以读取栅格和矢量数据，同时它的强大之处还在于下面几个方面，第一可以进行矢量图形之间的一些常用操作，比如：求交，求并，缓冲区等等。第二可以进行投影和坐标转换。如果使用GDAL默认的编译方式，那么上述的两个非常强大的功能您将不能使用，因为GDAL这两大功能是基于另外的两个开源库GEOS（Geometry Engine, Open Source）库和PROJ4库来实现的。下面对这两大库分别做一个简单的说明，以及如何修改编译文件，让GDAL能够拥有这两大功能。

## 1、集成GEOS

关于GEOS库的说明，网上有很多，同时在GEOS的官网[http://geos.osgeo.org](http://geos.osgeo.org/)有详细的说明，简单的来说，GEOS提供了OGC规范中简单几何要素对象操作的C++语言的实现。在地理信息系统领域，拓扑模型是重要的，其计算方法简单但是很难得以实现。使得GEOS不同于其他项目的也正是“空间谓词”与“空间操作”。空间谓词是比较两个空间对象并返回一个布尔变量值作为结果，它表明了存在于两个空间对象之间特殊的关系。比如典型的空间谓词有Contains(), Intersects(), Touches(), andCrosses()函数等。GEOS项目中对该些函数的实现是异常强壮的，即使是奇异几何对象或是临时的坐标系统运算也不能使其运算不正常或计算错误。目前绝大多数的商业软件仍然在最基础的空间谓词处理上相对成熟，这正是GEOS项目的重要意义。“空间操作”则主要是对两个几何对象进行计算并且返回一个新的几何实体。比较典型的操作函数如Difference(), Union()以及Buffer()等。GEOS中的操作算法已经被广泛的经过了测试。GEOS类库被各类开源空间信息软件项目广泛应用，使用GEOS，它们可以基于最新的规范的几何实体来完成，同时也拥有了复杂空间方法的实现。

关于GEOS的说明和编译，后面会单独写一篇文件进行介绍，这里假设已经下载的是编译好的GEOS库。

首先使用记事本或者其他的文本编辑器打开GDAL源代码目录下的nmake.opt文件，找到“# Uncomment for GEOS support”这句，大概在405行左右，将下面三行代码：

#GEOS\_DIR=C:/warmerda/geos

#GEOS\_CFLAGS =-I$(GEOS\_DIR)/capi -I$(GEOS\_DIR)/source/headers -DHAVE\_GEOS

#GEOS\_LIB     =$(GEOS\_DIR)/source/geos\_c\_i.lib

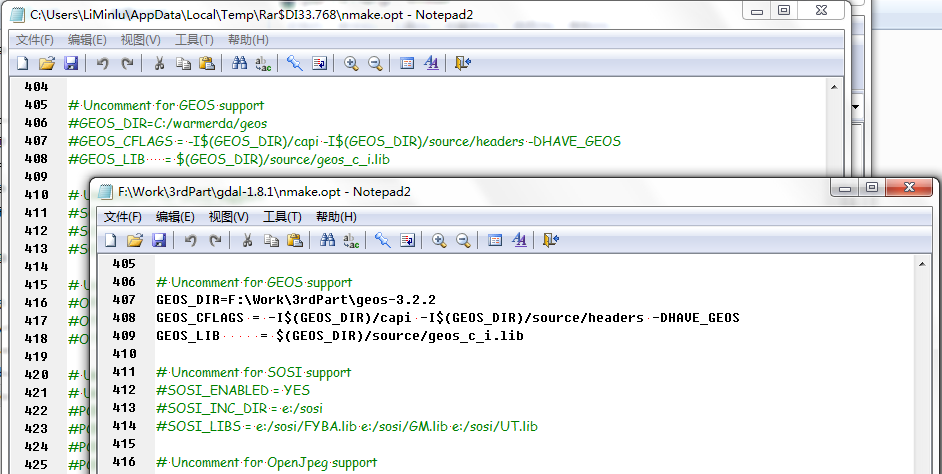
修改为：

GEOS\_DIR=F:\Work\3rdPart\geos-3.2.2

GEOS\_CFLAGS =-I$(GEOS\_DIR)/capi -I$(GEOS\_DIR)/source/headers -DHAVE\_GEOS

GEOS\_LIB     = $(GEOS\_DIR)/source/geos\_c\_i.lib

其中F:\Work\3rdPart\geos-3.2.2是我本机的GEOS存放的主目录，后面两行设置的是GEOS的头文件目录和lib文件路径。设置好后保存即可。对比结果如下图如下：



保存完nmake.opt之后，按照第一步中的编译方式进行编译即可。编译后的GDAL就将会支持图形之间的操作等处理。函数主要是在OGR库中，后面会在OGR库中进行详细的介绍说明。编译后，千万别忘记将geos\_c.dll文件拷贝到gdal18.dll的同级目录下，否则会提示你找不到geos\_c.dll文件。

## 2、集成Proj4

Proj4是一套开源的坐标投影转换类库，它可以完成在两套不同制图投影系统之间的转换，同样不同的椭球体或大地基准面之间也可以成功的完成转换。GDAL中用到的坐标转换，投影转换，几何纠正，正射纠正等算法，都离不开坐标转换，也就是说要使用这些算法，必须有proj4库的支持才行。同GEOS库的配置方法，在nmake.opt文件中，找到proj4库的位置，大概在352行左右。将下面的三行代码：

#PROJ\_FLAGS =-DPROJ\_STATIC

#PROJ\_INCLUDE =-Id:\projects\proj.4\src

#PROJ\_LIBRARY =d:\projects\proj.4\src\proj\_i.lib

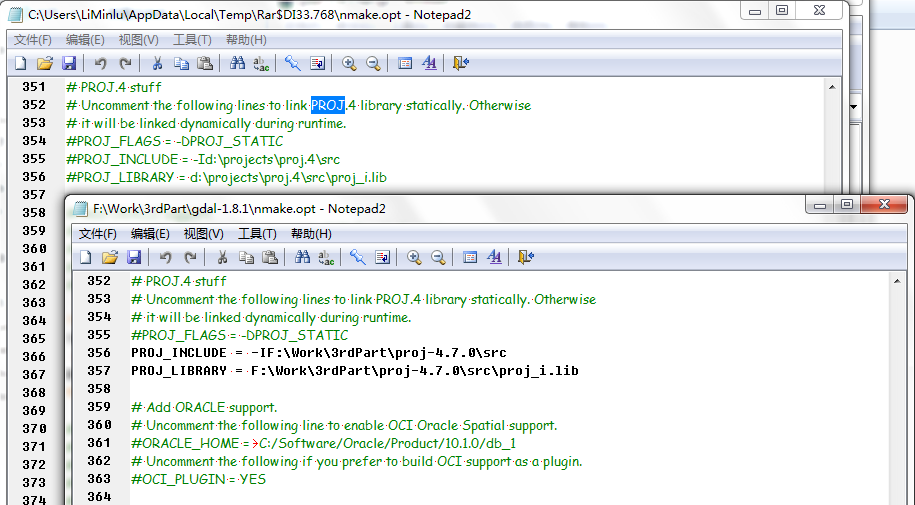
修改为：

#PROJ\_FLAGS =-DPROJ\_STATIC

PROJ\_INCLUDE =-IF:\Work\3rdPart\proj-4.7.0\src

PROJ\_LIBRARY =F:\Work\3rdPart\proj-4.7.0\src\proj\_i.lib

其中第一行表示是否使用静态链接的方式，第二行的路径表示，proj库存放的位置，第三行为proj库的lib文件所在路径。修改后保存即可，对比结果如下图如下：



同GEOS库一样，保存完nmake.opt之后，按照第一步中的编译方式进行编译即可。对于Proj库的使用后面会在有一篇文章对其做一个简单的介绍说明。编译后，同样千万别忘记将proj.dll文件拷贝到gdal18.dll的同级目录下，否则会提示你找不到proj.dll文件。

## 3、集成HDF数据读取

通过上面GEOS和PROJ库的介绍，相信对gdal的配置文件，nmake.opt有一个比较初步的了解了吧，那么下面对于使用GDAL支持hdf数据的读取也是同样，先下载好hdf4和hdf5的库，我用的是HDF4.2.6和HDF5-1.8.7两个库，在hdf的官方网上有编译好的库，直接下载编译好的库即可，对于hdf库的编译，我没有进行编译过，应该和其他的开源库都是差不多吧。同时官网提供了32位的库和64位的库，这里都是按照32位的库进行介绍，后面会有一个gdal的64位库的编译介绍。

将下载好的HDF4.2.6和HDF5-1.8.7两个库解压，然后修改nmake.opt文件中的278行左右，代码如下：

# Uncomment thefollowing and update to enable NCSA HDF Release 4 support.

#HDF4\_PLUGIN = NO

#HDF4\_DIR =       D:\warmerda\HDF41r5

#HDF4\_LIB =        /LIBPATH:$(HDF4\_DIR)\lib Ws2\_32.lib

# Uncomment thefollowing and update to enable NCSA HDF Release 5 support.

#HDF5\_PLUGIN = NO

#HDF5\_DIR =       c:\warmerda\supportlibs\hdf5\5-164-win

#HDF5\_LIB =        $(HDF5\_DIR)\dll\hdf5dll.lib

修改为下面的代码：

# Uncomment thefollowing and update to enable NCSA HDF Release 4 support.

HDF4\_PLUGIN = NO

HDF4\_DIR =          F:\Work\3rdPart\HDF4.2.6\_win\_x86

HDF4\_LIB =  $(HDF4\_DIR)\dll\hd426m.lib$(HDF4\_DIR)\dll\hm426m.lib \

$(HDF4\_DIR)\lib\hd426.lib$(HDF4\_DIR)\lib\hm426.lib Ws2\_32.lib

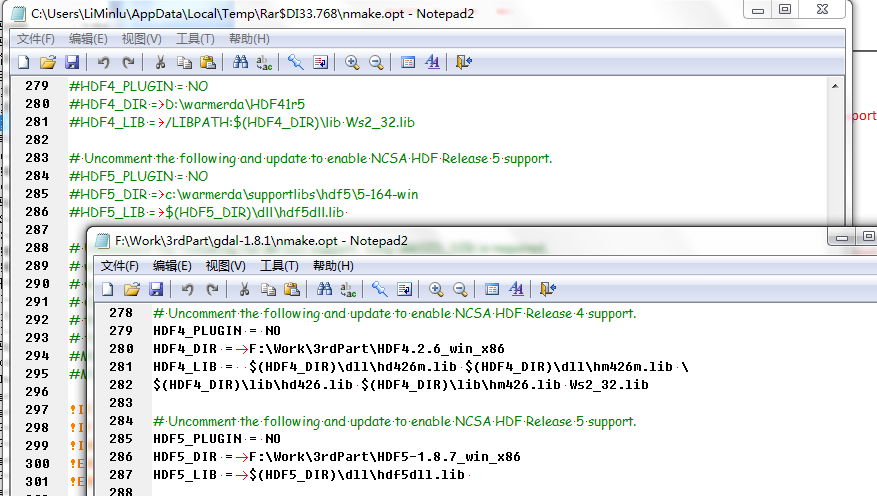
# Uncomment thefollowing and update to enable NCSA HDF Release 5 support.

HDF5\_PLUGIN = NO

HDF5\_DIR =          F:\Work\3rdPart\HDF5-1.8.7\_win\_x86

HDF5\_LIB = $(HDF5\_DIR)\dll\hdf5dll.lib

对比代码如下图：



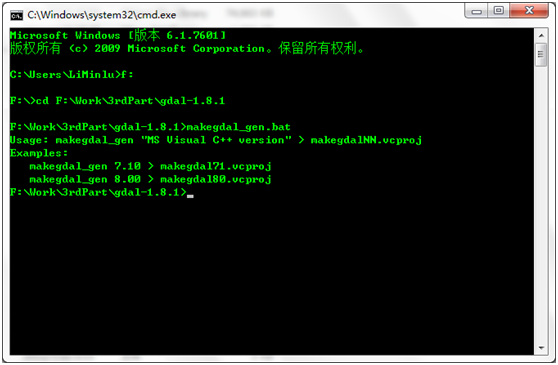
保存，然后编译gdal即可，同时将hdf库中的dll文件夹下的dll文件拷贝到gdal18.dll的同级目录下。

# 三、其他方面

## 1、makegdal\_gen.bat使用

对于makegdal\_gen.bat的作用，在上一篇文章中已经进行了介绍，下面对怎么使用该文件生成VS的工程文件做一个说明。

首先打开cmd命令行窗口，使用cd命令切换到GDAL源代码目录，然后输入makegdal\_gen.bat回车，会得到该工具的一个简单实用帮助，如下图所示：



该工具的使用方法是带有命令行参数的一个批处理工具，（在后面对GDAL工具集的介绍中会对带有参数的命令行程序，以及编写带有命令行的程序有一个比较详细的说明）。通过上图可以看出该工具的基本语法是：

makegdal\_gen 7.10 >makegdal71.vcproj

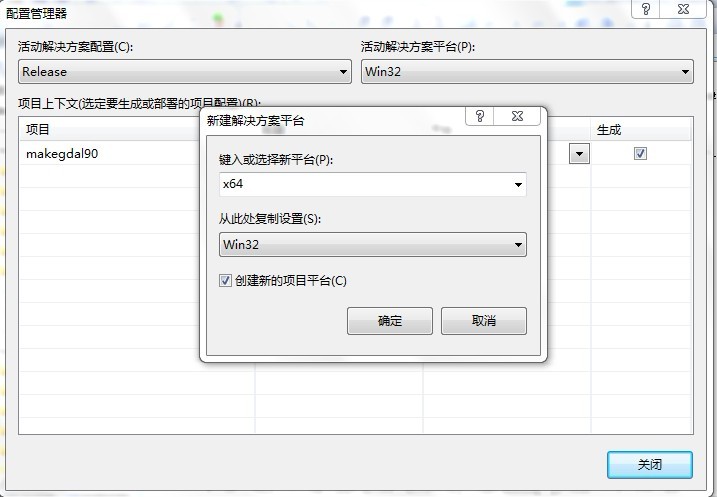
makegdal\_gen 8.00 >makegdal80.vcproj

通过上面的示例可以看出，该工具的命令行参数分别是，首先是VS的版本号，具体版本号参考本文第一小节，然后跟一个大于号“>”，最后是输出的VS的工程的名字。那么现在我要使用该命令行生成一个VS2008版本的工程文件，我可以输入下面的命令，然后回车即可：

makegdal\_gen 9.00 >makegdal90.vcproj

## 2、编译64位系统下的GDAL

对于GDAL的64位系统的编译，基本和32位系统的编译一样，首先在VS的工程中，打开配置管理器，然后再活动解决方案平台的下拉列表中选择新建，然后弹出，新建解决方案平台对话框，选择新平台为x64（需要在安装VS的时候安装64位的编译环境），然后点击确定即可。最后在VS中选择X64进行编译即可。如下图所示：



对于使用cmd命令行编译，基本同本文开始，不一样的只有，在开始菜单选择的不是“Visual Studio 2008 命令提示”而是“Visual Studio 2008 x64 兼容工具命令提示”，剩下的编译步骤跟前面的一样。

在编译开始之前，还需要打开nmake.opt文件，找到131行处的“#WIN64=YES”，将前面的“#”去掉，保存，然后开始编译。如果就这样编译过去的话，那么恭喜你，如果不能顺利编译过去的，那么需要按照下面的步骤进行一点点设置。

1：在GDAL目录下的nmake.opt文件中，找到SYM\_PREFIX的定义，应该在438行左右

将SYM\_PREFIX=\_ 改为SYM\_PREFIX= 就是将最后的下划线去掉

2：在GDAL目录下的makefile.vc文件中，找到46行左右的代码，如下：

BASE\_INCLUDE =/INCLUDE:\_GDALSimpleImageWarp@36 \

  /INCLUDE:\_GDALReprojectImage@48 \

  /INCLUDE:\_GDALComputeMedianCutPCT@32 \

  /INCLUDE:\_GDALDitherRGB2PCT@28 \

  /INCLUDE:\_OCTNewCoordinateTransformation@8$(VB6\_SAFEARRAYSYM)

修改为：

BASE\_INCLUDE =/INCLUDE:$(SYM\_PREFIX)GDALSimpleImageWarp \

  /INCLUDE:$(SYM\_PREFIX)GDALReprojectImage \

  /INCLUDE:$(SYM\_PREFIX)GDALComputeMedianCutPCT\

  /INCLUDE:$(SYM\_PREFIX)GDALDitherRGB2PCT \

 /INCLUDE:$(SYM\_PREFIX)OCTNewCoordinateTransformation $(VB6\_SAFEARRAYSYM)

就是将后面的@开始，后面的数字删除。现在开始编译吧，呵呵。祝你成功！此处可以参考我的博文：<http://liminlu0314.blog.163.com/blog/static/111800520106223508717/>

## 3、GDAL1.8后打开中文路径失败问题

此处可以参考我的博文：<http://blog.csdn.net/liminlu0314/article/details/6610069>，在此不再赘述。

[GDAL源码剖析（三）之Swig编译和帮助文档生成](http://blog.csdn.net/liminlu0314/article/details/6945452)

http://blog.csdn.net/liminlu0314/article/details/6945452

**一、Swig编译**

**1、Swig介绍**

SWIG全称是Simplified Wrapper and Interface Generator，官方网站：http://www.swig.org/。SWIG是个帮助使用C或者C++编写的软件能与其它各种高级编程语言进行嵌入联接的开发工具。SWIG能应用于各种不同类型的语言包括常用脚本编译语言例如Perl, PHP, Python, Tcl, Ruby and PHP。支持语言列表中也包括非脚本编译语言，例如C#, Common Lisp (CLISP, Allegro CL, CFFI, UFFI), Java, Modula-3, OCAML以及R，甚至是编译器或者汇编的计划应用（Guile, MzScheme, Chicken）。SWIG普遍应用于创建高级语言解析或汇编程序环境，用户接口，作为一种用来测试C/C++或进行原型设计的工具。SWIG还能够导出XML或Lisp s-expressions格式的解析树。SWIG可以被自由使用，发布，修改用于商业或非商业中。[摘自SWIG官网http://www.swig.org/translations/chinese/index.html]。  
下载安装Swig的时候注意下载Swigwin（我下的是swigwin-2.0.4.zip），不要下载源代码，否则不能在windwos下用。下载后解压，将swigwin-2.0.4的解压目录也添加到环境变量Path中去，否则会出现一些些该配置文件的麻烦。检验swig是否成功设置到环境变量Path中的最简单的方式就是在运行中输入swig后回车，如果提示windows找不到swig，那么说明没有设置成功；如果出现一个黑屏一闪而过，那么说明你设置成功了。

**2、编译C#版本GDAL**

首先，打开nmake.opt文件，找到SWIG=swig.exe这一句，假如没有将swig的目录添加到环境变量中，那么将这句后面的swig.exe修改为swig.exe的全路径，如F:\Work\3rdPart\swigwin-2.0.4\swig.exe。如果设置了环境变量，那么就不需要进行修改了。

然后按照第二篇中的使用cmd命令编译GDAL的方式来进行编译，打开“Visual Studio 2008命令提示”并定位到GDAL源代码目录，然后依次执行下面三行命令：

**[python]** [view plaincopyprint?](http://blog.csdn.net/liminlu0314/article/details/6945452)

1. nmake /f makefile.vc
2. nmake /f makefile.vc install
3. nmake /f makefile.vc devinstall

执行完之后会在GDAL\_HOME的目录下生成编译好的dll和exe文件以及include、lib等文件夹。接下来，使用cd命令，进入swig\csharp文件夹中：

**[python]** [view plaincopyprint?](http://blog.csdn.net/liminlu0314/article/details/6945452)

1. cd swig\csharp
2. #nmake /f makefile.vc interface     #这句话暂时不需要
3. nmake /f makefile.vc
4. nmake /f makefile.vc install

执行完上面两句后，会在csharp文件夹下生成8个dll文件，并将这8个dll文件拷贝到GDAL输出目录下的csharp文件夹中。然后就可以在C#工程中引用这几个dll了，需要注意的是，在使用这几个dll的时候需要将gdal18.dll以及其依赖的其他dll都要拷贝到同一个目录中才能正常运行。

**3、编译Java版本GDAL**

首先需要JDK环境，没有的可以下载安装一个。然后打开nmake.opt文件，找到大概86行左右的位置：

**[python]** [view plaincopyprint?](http://blog.csdn.net/liminlu0314/article/details/6945452)

1. JAVA\_HOME = C:\Program Files\Java\jdk1.

将后面的路径修改为Python的安装路径后保存。使用cd命令将命令行定位到swig\python目录后，依次输入：

**[python]** [view plaincopyprint?](http://blog.csdn.net/liminlu0314/article/details/6945452)

1. python setup.py build
2. python setup.py install

执行完上述命令后，会在python的site-packages目录看到多了gdal和ogr的文件以及一个osgeo的文件夹。我本机的目录是C:\Python26\Lib\site-packages。然后将编译出来的gdal18.dll以及它依赖的所有的dll都拷贝到site-packages文件夹中的osgeo文件夹中，最后就可以在Python中使用imort gdal来使用GDAL了。测试Python版本的gdal是否安装成功，可以使用下面命令：

**[python]** [view plaincopyprint?](http://blog.csdn.net/liminlu0314/article/details/6945452)

1. **from** osgeo **import** gdal
2. **from** osgeo.gdalconst **import** \*
3. dataset=gdal.Open("F:\Work\Data\envi.img",GA\_ReadOnly)
4. dataset.GetDriver().ShortName
5. #'ENVI' #图像格式
6. dataset.GetProjectionRef()
7. #'PROJCS["unnamed",GEOGCS["NAD83",DATUM["North\_American\_Datum\_1983",SPHEROID["GRS 1980",6378137,298.257222101,AUTHORITY["EPSG","7019"]],TOWGS84[0,0,0,0,0,0,0],AUTHORITY["EPSG","6269"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9108"]],AUTHORITY["EPSG","4269"]],PROJECTION["Transverse\_Mercator"],PARAMETER["latitude\_of\_origin",30],PARAMETER["central\_meridian",-84.16666666666667],PARAMETER["scale\_factor",0.9999],PARAMETER["false\_easting",2296587.926509186],PARAMETER["false\_northing",0],UNIT["Foot (International)",0.3048]]' #图像投影信息

注意，上面命令中开头为#号的表示的是Python输出的。如果能输出上面的信息，那么GDAL for Python就安装成功了。

**二、开发帮助文档生成**

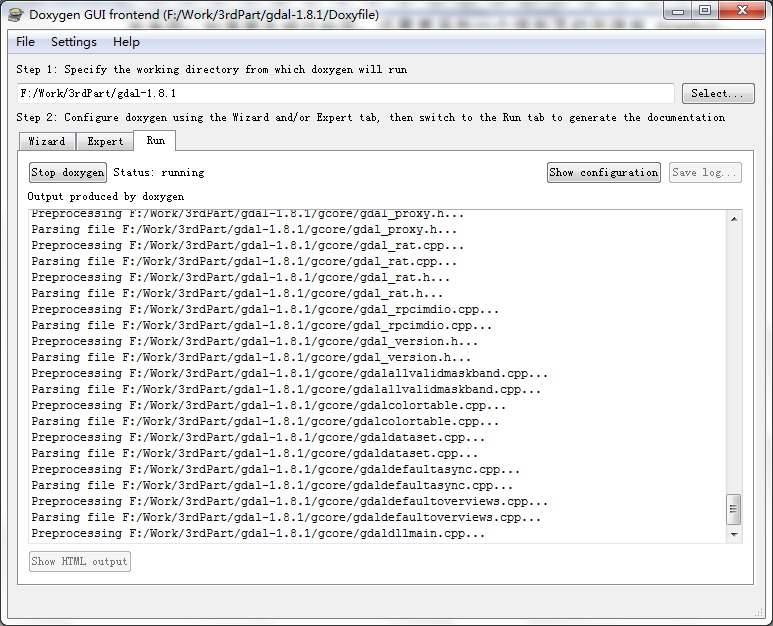
**1、Doxygen介绍**

以下内容摘自维基百科，地址http://zh.wikipedia.org/wiki/Doxygen。   
Doxygen 是一个 C++, C, Java, Objective-C、Python、IDL (CORBA 和 Microsoft flavors)、Fortran、VHDL、PHP、C#和D语言的文档生成器。可以运行在大多数类Unix系统，以及Mac OS X操作系统和Microsoft Windows 。 初始版本的Doxygen借鉴了一些老版本DOC++的代码；随后，Doxygen源代码由Dimitri van Heesch重写。  
Doxygen是一个编写软件参考文档的工具。 该文档是直接写在代码中，因此比较容易保持更新。 Doxygen 可以交叉引用文档和代码，使文件的读者可以很容易地引用实际的代码。  
KDE 使用Doxygen作为其部分文档且 KDevelop 具有内置的支持。 Doxygen的发布遵守GNU General Public License，并且是自由软件。  
需要注意的是在使用doxygen的时候，会自动生成一些类图，以及函数调用关系图，如果要生成这些图，还需要另外一个很牛叉的开源库Graphviz。Graphviz （Graph Visualization Software的缩写）是一个由AT&T实验室启动的开源工具包，用于绘制DOT语言脚本描述的图形。它也提供了供其它软件使用的库。Graphviz是一个自由软件，其授权为Common Public License。其Mac版本曾经获得2004年的苹果设计奖。更多关于Graphviz的介绍参考其官方网站：http://www.graphviz.org/。

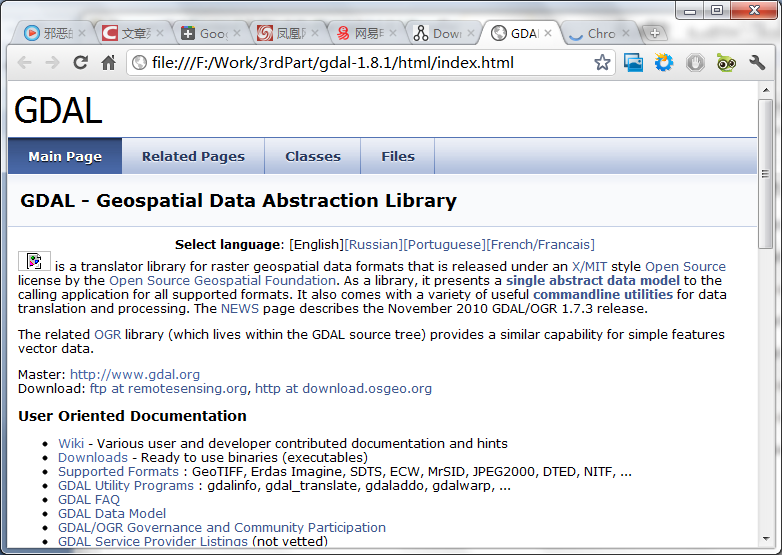
**2、生成GDAL开发帮助文档**

首先下载安装Doxygen和Graphviz，安装完之后最好将Doxygen和Graphviz的bin目录添加到系统环境变量Path中去。负责要设置一些参数，比较麻烦，还是放到Path中方便。

安装完Doxygen后，会在开始菜单中有个doxywizard.exe的程序，打开，然后在File->Open…菜单中选择GDAL源代码目录下的Doxyfile文件，然后，切换到Run标签，点击Run Doxygen按钮，接下来就会自动提取源代码中的注释生成一份gdal的帮助文档，默认的输出目录是GDAL目录下的html目录。等待生成结束后，点击左下角的Show Html Output后会打开生成的帮助文档。如下图：



再次，基本上GDAL的帮助文档生成完成，打开后会发现和GDAL的官方网站一模一样。但是还有个问题，这里只是生成的是GDAL的帮助文档，没有OGR的帮助文档，同样按照上面的步骤，打开GDAL目录下的OGR文件夹下的Doxyfile。然后点击生成，生成的目录默认为ogr文件夹下有个html文件夹，将该文件夹重命名为ogr，然后整个拷贝到上一层的html中，同时将GDAL目录中的doc文件夹中的文件除dox文件以外的文件全部拷贝到html文件夹中。  
最后，还记得编译GDAL后生成的html文件夹吗？将这两个文件夹中的内容进行合并，然后你就得到了一份完完整整的GDAL的帮助文档。首页是html文件夹下的index.html。如下图：



## 3、Doxygen脚本配置以及代码注释书写规范

对于Doxygen的脚本配置，可以打开doxywizard后，在Wizard和Expert 标签中在每一项中，鼠标移动到上面，在左下角的试图区域会显示很详细的帮助信息（英文的，不是很难），这里有篇说明文档《 Doxygen配置使用指南》，http://read.pudn.com/downloads151/doc/653112/Doxygen\_Using\_Manual.pdf。文档中包含了所有的东西，很不错的东西。

[GDAL源码剖析（四）之命令行程序说明一](http://blog.csdn.net/liminlu0314/article/details/6978589)

http://blog.csdn.net/liminlu0314/article/details/6978589

## 一、GDAL工具通用命令

下面的工具主要参考的GDAL官方网站中提供的帮助文档说明，此外还有我的一些经验，GDAL官方具体地址为：<http://gdal.org/gdal_utilities.html>。

在所有的GDAL工具集中都会支持下面的通用命令行参数，其形式一般是以两个减号（--）开始，下面详细介绍：

### 1.             –version

输出GDAL的版本信息，即版本号。

### 2.             --formats

输出GDAL支持的所有图像格式说明。包括只读和读写。格式支持描述如下：“ro”是只读驱动；“rw”是读写驱动（比如支持CreateCopy方法）；“rw+”是读写和更新驱动（比如支持Create方法），支持所有的读写更新操作。

### 3.             --format format

输出GDAL单个格式驱动的细节信息。格式名需要是在--formats 后列出所要输出的格式名。比如GTiff，HFA,PCISK等。

### 4.             --optfile file

读取指定名称的文件并把其中的内容当成参数传入命令行列表。如果行首以#开头的行将被忽略。多字组成的参数（即中间有空格隔开的参数）需要用双引号来保正其为单一的参数。

### 5.             --config key value

设置配置，把指定键设置为某个值，从而不必把他们设置为环境变量。一些命令参数键是GDAL\_CACHEMAX(用于缓存的内存有多少M)以及GDAL\_DATA（gdal的数据路径）。比如在GDAL1.8之后，经常会发现打不开中文路径的文件，那么可以用这个来设置，具体为“—configGDAL\_FILENAME\_IS\_UTF8 NO”。同时对于每一种驱动都会有各自的配置，具体参考各个驱动的说明。更多的配置选项参考该网址：<http://trac.osgeo.org/gdal/wiki/ConfigOptions>。

### 6.             --debug value

控制调试信息的打印输出。ON值表示允许调试信息输出，OFF值表示不要输出调试信息。

### 7.             --help-general

输出各个工具的命令行参数帮助信息。不同的命令输出的内容不同。

下面还有一些通用的命令，是用来创建文件来使用的。创建不同的格式需要的参数都是不相同的，尤其是在特殊的情况下，比如创建的Erdas的img格式需不需要使用压缩等特殊的需求。这些参数一般使用一个减号（-）开始。下面对这些参数进行一个简单的说明。

### 8.             -of format

选择要创建新的文件的格式。这个格式被指定为类似GTiff（GeoTIFF格式）或者HFA（ERDAS格式）。所有的支持格式列表可以用--formats 参数列出来。但是只有格式列表“(rw)”可以被写入和创建。许多工具如果没有指定，默认是创建GeoTIFF格式的文件。文件扩展名不会自动添加，如果没有指定文件名的后缀名，gdal一般不会添加任何扩展名。各个工具的命令行参数帮助信息。不同的命令输出的内容不同。

### 9.             -co NAME=VALUE

创建文件选项，许多格式会有一个或者更多的创建参数来控制文件创建的细节。比如GeoTIFF或者Erdas的img格式可以用创建参数控制压缩，或者控制是否用分片还是分带来进行存储。

可以使用的创建参数根据格式驱动不同而不同。而一些简单的格式根本就没有创建参数。虽然某个格式可以用"--format <format>"参数列出所有可用的参数列表，但是更详细的信息可以在格式介绍网页中查到。对于不同的文件格式，请参考对应文件格式说明网页。

### 10.       -a\_srs SRS

指定输出文件的投影信息（坐标系统）。输出各个工具的命令行参数帮助信息。不同的命令输出的内容不同。有几个工具(如gdal\_translate、gdalwarp)可以在命令行中通过类似-a\_srs(分配输出SRS)、-s\_srs(源SRS)、-t\_srs(目标SRS)来指定坐标系统。这些工具允许以一系列格式定义坐标系统（SRS就是空间参考系统spatial reference system）。SRS通常可以使用下面几种方式来指定：

* **NAD27/NAD83/WGS84/WGS72:**这些常见的地理坐标系统可以通过名字来直接使用。
* **EPSG:**n：坐标系统（投影或者地理坐标）可以通过EPSG码来选择。例如EPSG 27700是英国国家网格。更多的EPSG坐标系统可以在GDAL数据文件gcs.csv和pcs.csv中找到（位于GDAL目录中的data文件夹中）。
* **PROJ.4定义：**一个PROJ4定义字符串可以用作坐标系统定义。例如“+proj=utm +zone=11 +datum=WGS84”。注意在命令行中要保持Proj4字符串在一起作为一个单独的参数（一般用双引号引起来）。
* **OpenGIS WKT字符串:** OpenGIS标准定义了一个文本格式来描述坐标系统作为简单要素规范的一个部分。这个格式是gdal中使用的坐标系统的内部工作格式。包含wkt坐标系统描述的文件的文件名可以被用来作为坐标系统参数，或者坐标系统元素本身也可以被用来作为命令行参数。
* **ESRI WKT字符串：**ESRI 在他们的ArcGIS产品（ArcGIS中的.prj文件）中使用了一种经过精简OGC WKT的格式，而且这个格式被用在一个和wkt相似的风格的文件中。但是文件名要被加以ESRI::前缀。比如"ESRI::NAD 1927 StatePlane Wyoming West FIPS 4904.prj"。
* **空间参考网址URLs：**可以使用一个空间参考的网址来指定，如：<http://spatialreference.org/ref/user/north-pacific-albers-conic-equal-area/>。
* **文件名：**可以使用一个包含WKT、Proj.4的字符串，或者XML/GML格式的坐标系统定义的文件。

## 二、GDAL工具说明

### 1.             gdalinfo 输出文件信息

用法：

gdalinfo[--help-general] [-mm] [-stats] [-hist] [-nogcp] [-nomd]

         [-noct] [-nofl] [-checksum] [-proj4][-mdd domain]\*

        [-sd subdataset] datasetname

参数说明：

gdalinfo程序输出gdal支持的栅格格式的一系列信息。

**-mm**

强制计算栅格每个波段的最大最小值。

**-stats**

读取和现实图像统计信息，如果指定该参数，将强制计算图像的统计信息，如各个波段的最大值、最小值、均值、标准差等。

**-hist**

输出所有波段的直方图信息。

**-nogcp**

禁止地面控制点（GCP）列表打印。这可能对大量的GCP的数据集来说是十分有用的。比如L1B AVHRR或者hdf4MODIS数据，这些数据包含了成千上万的地面控制点。

**-nomd**

禁止元数据打印，一些数据集可能包含极多的元数据字符串。

**-noct**

禁止输出颜色表。

**-checksum**

强制计算数据集中所有波段的checksum。

**-mdd domain**

输出指定区域的元数据信息。

**-nofl**

仅显示文件列表中的第一个文件信息。GDAL1.9.0开始支持该参数。

**-sd** subdataset

如果输入的数据集包含几个子数据集，那么将使用指定的数字来替代（从1开始）完整的子数据集名称。GDAL1.9.0开始支持该参数。

**-proj4**

输出文件的坐标系统按照PROJ.4类型的字符串输出。GDAL1.9.0开始支持该参数。

gdalinfo同时会输出如下的信息（如果有的话）：

* 当前文件的格式驱动信息
* 栅格数据大小(行列数)
* 文件的坐标系统(OGC WKT形式)
* 图像关联到地理的转换参数(当前不包含旋转系数)
* 地理上的边界坐标，如果可能的话还有基于经纬度的完整的地理转换参数（如果是GCPs就没有）
* 地面控制点(GCPs)
* 所有的（包括子栅格的元数据）文件元数据
* 波段数据类型
* 波段颜色信息（RGB,Gray等）
* 波段颜色表信息
* 波段瓦片大小（文件块大小）
* 波段描述
* 波段最大最小值（已经经过计算的）
* 波段CheckSum值（已经经过计算的）
* 波段无意义值（NODATA值）
* 波段可获得的略缩图分辨率
* 波段单位类型(如：波段的高程是米制还是英制)
* 波段的假颜色列表

举例：

gdalinfoF:/Work/Data/utm.tif

Driver:GTiff/GeoTIFF

Sizeis 512, 512

CoordinateSystem is:

PROJCS["NAD27/ UTM zone 11N",

    GEOGCS["NAD27",

       DATUM["North\_American\_Datum\_1927",

            SPHEROID["Clarke1866",6378206.4,294.978698213901]],

        PRIMEM["Greenwich",0],

       UNIT["degree",0.0174532925199433]],

   PROJECTION["Transverse\_Mercator"],

   PARAMETER["latitude\_of\_origin",0],

   PARAMETER["central\_meridian",-117],

    PARAMETER["scale\_factor",0.9996],

   PARAMETER["false\_easting",500000],

    PARAMETER["false\_northing",0],

    UNIT["metre",1]]

Origin= (440720.000000,3751320.000000)

PixelSize = (60.000000,-60.000000)

CornerCoordinates:

UpperLeft  ( 440720.000, 3751320.000) (117d38'28.21"W, 33d54'8.47"N)

LowerLeft  ( 440720.000, 3720600.000) (117d38'20.79"W, 33d37'31.04"N)

UpperRight (  471440.000, 3751320.000)(117d18'32.07"W, 33d54'13.08"N)

LowerRight (  471440.000, 3720600.000)(117d18'28.50"W, 33d37'35.61"N)

Center      ( 456080.000, 3735960.000) (117d28'27.39"W, 33d45'52.46"N)

Band1 Block=512x16 Type=Byte, ColorInterp=Gray

### 2.             gdal\_translate 格式转换

用法：

gdal\_translate[--help-general]

       [-ot{Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/

             CInt16/CInt32/CFloat32/CFloat64}][-strict]

       [-of format] [-b band] [-mask band][-expand {gray|rgb|rgba}]

       [-outsize xsize[%] ysize[%]]

       [-unscale] [-scale [src\_min src\_max[dst\_min dst\_max]]]

       [-srcwin xoff yoff xsize ysize][-projwin ulx uly lrx lry]

       [-a\_srs srs\_def] [-a\_ullr ulx uly lrxlry] [-a\_nodata value]

       [-gcp pixel line easting northing[elevation]]\*

       [-mo "META-TAG=VALUE"]\* [-q][-sds]

       [-co "NAME=VALUE"]\* [-stats]

       src\_dataset dst\_dataset

参数说明：

gdal\_translate工具可以用来在不同格式间转换栅格数据。同时还可以做一些诸如提取子栅格、重采样和数据类型转换等操作。

**-ot**: type

指定输出波段的数据类型。

**-strict**:

对于这转换过程中出现丢失数据等错误直接报错，不进行忽略处理。之前的参数叫**-not\_strict**，表示对其进行宽大处理，不报错。

**-of** format:

选择输出格式。默认是[GeoTiff](http://wiki.woodpecker.org.cn/moin/GeoTiff)（GTiff）。注意，指定的时候用格式简称。

**-b** band:

选择一个波段来输出。波段号从1开始编号。多个 **-b** 参数可以用于选择输出某几个波段或者重新对波段进行排序。从GDAL1.8.0开始，波段可以设置为“mask,1”（或者直接mask）来讲输入数据集的第一个波段作为一个mask波段来使用。

**-mask** band:

（从GDAL1.8.0开始）(GDAL >= 1.8.0)选择一个输入波段来作为创建输出数据的掩码波段。波段数是从1开始，band可以设置为none，用来避免复制整个输入数据集作为掩码，否则在默认情况下（“auto”），除非掩码是一个alpha通道，或者使用参数-b mask来进行输出，参数band也可以设置为“mask,1”（或者直接mask）来讲输入数据集的第一个波段作为一个mask波段来使用。

**-expand** gray|rgb|rgba:

（从GDAL1.6.0开始）将带有颜色表信息的单波段文件展开为RGB三波段文件或者RGBA四波段文件。对于输出格式为JPEG，JPEG2000，MrSID，ECW等不支持颜色表的数据来说很有用。灰度值（从GDAL1.7.0开始）也可以使用颜色表展开为一个数据集，但输出文件中仅仅包含灰度级别的一个索引。

**-outsize** xsize[%] ysize[%]:

设置输出文件的大小。大小以象元为单位，除非用“%”来标记，这时，表示的是输出为输入图层大小的百分比。

**-scale** [src\_min src\_max [dst\_mindst\_max]]:

重新组织输入象元的值。将它们从src\_min~src\_max范围内缩放到dst\_min ~ dst\_max范围内。如果省略，输出范围将为0~255。输入范围将由源数据自动计算。

**-unscale**:

大概意思是，不对波段中的数据进行缩放转换，在使用**-ot**设置输出文件类型时这个参数往往是很有用的。（这个参数的英文有点绕口，不好翻译，有知道的同学麻烦告诉我一下）。

**-srcwin** xoff yoff xsize ysize:

选择一个取值窗口，通过该窗口在原图像中的行列位置来拷贝数值。

**-projwin** ulx uly lrx lry:

选择一个取值窗口，通过该窗口在原图像中地理坐标范围来拷贝数据（类似srcwin）。参数中的四个值，使用的投影坐标。

**-a\_srs** srs\_def:

给输出文件投影强制指定坐标系。srs\_def可以是任何常用的GDAL/OGR格式的投影信息，如：WKT、Proj4、EPSG:n 或者一个包含着wkt的文件的文件名。

**-a\_ullr** ulx uly lrx lry:

强制指定输出文件的空间转换边界范围（图像的四至范围）。而将原图像的四至范围忽略掉。

**-a\_nodata** value:

指定一个无意义值到输出波段。从GDAl1.8.0开始，可以设置为none来使用原文件中的nodata值作为输出文件的nodata值。

**-mo** "META-TAG=VALUE":

如果可以，给输出数据设置一个元数据的键和其对应的值。

**-co** "NAME=VALUE":

通过一个创建参数来指定输出格式特殊创建要求。多个-co 参数可以组合起来使用。创建参数可以参考各个数据格式本身说明。

**-gcp** pixel line easting northingelevation:

添加指定地面控制点到输出数据集。这个选项可以被多次使用，以提供一系列的地面控制点GCPs 。GCP格式为：列号 行号 横坐标 纵坐标 高程值。

**-q**:

安静模式，不输出进度信息以及其他非错误信息。

**-sds**:

拷贝文件中所有子数据集到各自的输出文件中。通常这个参数用在HDF或者OGDI这样有子数据集的格式中。

**-stats**:

强制计算（重新计算）数据的统计信息。自GDAL1.8.0开始支持该参数。

src\_dataset:

输入数据集名称，可以是文件名，或者是一个多数据集文件中的一个子数据集的URL地址（比如HDF数据集中的一个子数据集）。

dst\_dataset:

输出文件名。

举例：

将utm.tif转换为一个以分块存储的GeoTiff文件。

gdal\_translate -of GTiff-co "TILED=YES" utm.tif utm\_tiled.tif

创建一个JPEG压缩的Tiff图像，同时使用内部掩码从一个RGBA数据集中。

gdal\_translate rgba.tifwithmask.tif -b 1 -b 2 -b 3 -mask 4 -co COMPRESS=JPEG -co PHOTOMETRIC=YCBCR--config GDAL\_TIFF\_INTERNAL\_MASK YES

创建一个RGBA图像从一个RGB数据中使用一个掩码。

gdal\_translatewithmask.tif rgba.tif -b 1 -b 2 -b 3 -b mask

### 3.             gdaladdo 建立金字塔

用法：

gdaladdo[-r {nearest,average,gauss,cubic,average\_mp,average\_magphase,mode}]

         [-ro] [-clean] [--help-general]filename levels

参数说明：

gdaladdo工具可以用于为大多数支持的格式建立或者重建金字塔。可以使用下面几种重采样算法中的一种来进行缩小重采样操作。

**-r** {nearest(default),average,gauss,cubic,average\_mp,average\_magphase,mode}:

指定重采样方法，分别是最邻近（默认），均值，高斯，立方卷积等。

**-ro**:

（从GDAL1.6.0开始有效），以只读的方式打开数据，来创建外部金字塔（特别对于GeoTIFF来说）。

**-clean**:

删除所有的金字塔。（从GDAL1.7.0开始有效）。

filename:

要建立金字塔的文件名。

levels:

要建立略缩图的层号的列表。 如果使用-clean选项是将被忽略。

模式，（从GDAL1.6.0开始有效）选择最常用的重采样方式。average\_mp 是不适合使用的， Average\_magphase用于复数数据空间的图像。Nearest 和 average 用于普通的图像。Nearest 使用最邻近采样（简单采样），它计算所有的有效值的均值来进行计算。Cubic 采样（从GDAL1.7.0开始有效）使用一个4x4的近似立方卷积内核。 Gauss 采样（从GDAL1.6.0开始有效）使用高斯内核计算。这种对于高对比度和图案边界比较明显的图像效果比较好。一般建议的采样比值是2,4,8,…，使用3x3重采样作为高斯采样的计算窗口。

gdaladdo将遵守正确NODATA\_VALUES元组（特殊的数据集元数据），因此，只有一个给定的的RGB三元组（在一个RGB图像的情况下）作为NODATA值，而不是每个波段有独立的NODATA值。

选择一个缩放级别值如2表示略缩图缩放程度是源图像每个维上分辨率的1/2。如果文件在所选缩放级别上已经存在略缩图，那么这个缩放级别上的缩略图将被重新计算并覆盖写入。

一些格式根本不支持金字塔。所以许多格式在文件以外以扩展名.ovr存储金字塔，TIFF就是如此。GeoTIFF格式直接把金字塔存储到原有的文件中。除非使用-ro标记来指定。在TIFF中创建金字塔可以通过用COMPRESS\_OVERVIEW配置参数进行压缩。所有GeoTIFF支持的压缩方法，可以在这里获得（如：--config COMPRESS\_OVERVIEW DEFLATE）。

大多数驱动也支持一个备用的略缩图格式（使用的是Erdas图像格式）。要使用这个备用格式使用USE\_RRD=YES 来设置参数。这样做会把GDAL程序创建的金字塔放到一个辅助的.aux文件中使得可以该金字塔可以直接在Erdas中使用或者也可以在ArcGIS中使用。关于如何使用GDAL创建Erdas格式的金字塔，请参考我的博文：<http://blog.csdn.net/liminlu0314/article/details/6127755>。

举例：

在所指定的TIFF文件内部创建金字塔：

gdaladdo -r average abc.tif 2 4 8 16

从一个ERDAS.IMG文件中创建一个外部的压缩的金字塔文件：

gdaladdo --config COMPRESS\_OVERVIEWDEFLATE erdas.img 2 4 8 16

为给定JPEG文件创建一个Erdas Imagine 格式金字塔：

gdaladdo --config USE\_RRD YES airphoto.jpg3 9 27 81

[GDAL源码剖析（四）之命令行程序说明二](http://blog.csdn.net/liminlu0314/article/details/7218543)

http://blog.csdn.net/liminlu0314/article/details/7218543

   接博客[GDAL源码剖析（四）之命令行程序说明一http://blog.csdn.net/liminlu0314/article/details/6978589](http://blog.csdn.net/liminlu0314/article/details/6978589)

其中有个nearblack,gdalbuildvrt工具，没有做说明，以后加上，里面可能有很多不足，望大家批评指正，谢谢。

### 1. gdalwarp 图像纠正

用法：

gdalwarp [--help-general] [--formats]

    [-s\_srs srs\_def] [-t\_srs srs\_def] [-to "NAME=VALUE"]

    [-order n | -tps | -rpc | -geoloc] [-et err\_threshold]

    [-refine\_gcps tolerance [minimum\_gcps]]

    [-te xmin ymin xmax ymax] [-tr xres yres] [-tap] [-ts width height]

    [-wo "NAME=VALUE"] [-ot Byte/Int16/...] [-wt Byte/Int16]

    [-srcnodata "value [value...]"] [-dstnodata "value [value...]"] -dstalpha

    [-r resampling\_method] [-wm memory\_in\_mb] [-multi] [-q]

    [-cutline datasource] [-cl layer] [-cwhere expression]

    [-csql statement] [-cblend dist\_in\_pixels] [-crop\_to\_cutline]

    [-of format] [-co "NAME=VALUE"]\* [-overwrite]

    srcfile\* dstfile

参数说明：

gdalwarp工具是一个图像镶嵌、重投影、和纠正的工具。程序可以重投影到任何支持的投影。

-s\_srs srs def:

设置原始空间参考，坐标系统是可以使用函数OGRSpatialReference.SetFromUserInput()调用的就行，包括EPSG PCS，PROJ4或者后缀名为.prf的wkt文本文件。

-t\_srs srs\_def:

设置目标空间参考，坐标系统是可以使用函数OGRSpatialReference.SetFromUserInput()调用的就行，包括EPSG PCS，PROJ4或者后缀名为.prf的wkt文本文件。

-to NAME=VALUE:

设置转换参数选项，具体选项参考函数[GDALCreateGenImgProjTransformer2()](http://blog.csdn.net/liminlu0314/article/details/7218543#94cd172f78dbc41d6f407d662914f2e3)支持的选项。

-order n:

多项式纠正次数（1到3），默认的多项式次数根据输入的GCP点个数自动计算。

-tps:

强制使用TPS（thin plate spline）纠正方法来纠正图像。

-rpc:

强制使用RPC参数纠正。

-geoloc:

强制使用Geolocation数组。（这个没用过，不太清楚）

-et err\_threshold:

指定变换的近似误差阈值，默认为0.125个像元大小（使用像元为单位）。

-refine\_gcps tolerance minimum\_gcps:

(GDAL >= 1.9.0) refines the GCPs by automatically eliminating outliers. Outliers will be eliminated until minimum\_gcps are left or when no outliers can be detected. The tolerance is passed to adjust when a GCP will be eliminated. Not that GCP refinement only works with polynomial interpolation. The tolerance is in pixel units if no projection is available, otherwise it is in SRS units. If minimum\_gcps is not provided, the minimum GCPs according to the polynomial model is used. （这个参数不太清楚，没用过）

-te xmin ymin xmax ymax:

设置输出文件的地理范围（在目标空间参考中）。

-tr xres yres:

设置输出图像的分辨率。

-tap:

(GDAL >= 1.8.0) (target aligned pixels) align the coordinates of the extent of the output file to the values of the -tr, such that the aligned extent includes the minimum extent.

-ts width height:

设置输出文件的宽高。如果宽或者高有一个为0，那么将自动计算一个值，注意-ts和-tr不能同时使用。

-wo "NAME=VALUE":

设置纠正选项。具体参考[GDALWarpOptions::papszWarpOptions](http://blog.csdn.net/liminlu0314/article/details/7218543#0ed77f9917bb96c7a9aabd73d4d06e08)的帮助文档。

-ot type:

指定输出波段的数据类型。

-wt type:

计算的数据类型。

-r resampling\_method:

重采样方式，主要有下面几种方式：

near:

最邻近采样方法（默认值，算法较快，但是质量较差）。

bilinear:

双线性内插采样。

cubic:

立方卷积采样。

cubicspline:

立方样条采样。

lanczos:

Lanczos 窗口辛克采样。

-srcnodata value [value...]:

设置输入波段的Nodata值，可以为不同的波段指定不同的值。。如果有多个值，就需要把他们用双引号括起来，以保持在命令参数中作为单一参数输入。掩膜值不会在内插中处理。

-dstnodata value [value...]:

设置输出波段的Nodata值。

-dstalpha:

创建一个Alpha波段在输出文件中。

-wm memory\_in\_mb:

设置纠正API使用的内存大小，以MB为单位。

-multi:

是否使用多线程纠正图像，多线程用来分块处理，同时在读取和写入图像均使用多线程技术。

-q:

不在控制台输出提示信息。

-of format:

输出文件格式，默认为GeoTiff。

-co "NAME=VALUE":

指定创建图像选项，具体参考不同的格式说明。

-cutline datasource:

使用使用OGR支持的矢量数据进行裁切图像。

-cl layername:

指定裁切矢量的图层名称。

-cwhere expression:

从裁切矢量中根据属性表查询指定的要素来裁切图像。

-csql query:

使用SQL语句来从裁切矢量的属性表中查询要素来裁切图像。

-cblend distance:

Set a blend distance to use to blend over cutlines (in pixels).（这个参数不太清楚，没用过）

-crop\_to\_cutline:

(GDAL >= 1.8.0) 使用矢量边界的外接矩形大小作为输出影像的范围。

-overwrite:

(GDAL >= 1.8.0) 如果结果数据存在，那么覆盖结果数据。

srcfile:

输入数据文件名（可以为多个，使用空格隔开）。

dstfile:

输出数据文件名。

如果输出文件已经存在，那么镶嵌到这个文件是可以的。但是数据的空间范围等信息不会被修改，如果要修改为新的数据的空间信息，那么需要使用-overwrite选项来覆盖原文件。

举例：

在此不再列举例子，后面会对gdalwarp做一个比较详细的介绍。

### 2. gdaltindex 创建一个栅格文件块索引Shp文件

用法：

gdaltindex [-tileindex field\_name] [-write\_absolute\_path] [-skip\_different\_projection] index\_file [gdal\_file]\*

参数说明：

这个程序创建一个shape文件，里面记录每一个输入的栅格文件的文件路径，以及栅格数据的落图文件。这个输出文件主要是为[MapServer](http://mapserver.org/) 来使用的。应该是创建栅格图像的四至范围矢量文件。

· 如果shape文件不存在会自己创建，如果存在，将被追加到存在的文件中。

· 默认的索引字段为location。

· 默认填写的是栅格文件名，如果指定了-write\_absolute\_path 选项，那么写入的是栅格文件的绝对路径。

· 如果指定-skip\_different\_projection参数，只用相同的投影的栅格文件会被插入到tileindex文件中。

· 生成一个简单的矩形多边形，其坐标系统和栅格的坐标系统相同。

举例：

gdaltindex doq\_index.shp doq/\*.tif

### 3. gdalbuildvrt 创建VRT数据

用法：

gdalbuildvrt [-tileindex field\_name] [-resolution {highest|lowest|average|user}]

             [-tr xres yres] [-tap] [-separate] [-allow\_projection\_difference] [-q]

             [-te xmin ymin xmax ymax] [-addalpha] [-hidenodata]

             [-srcnodata "value [value...]"] [-vrtnodata "value [value...]"]

             [-input\_file\_list my\_liste.txt] [-overwrite] output.vrt [gdalfile]\*

参数说明：

从指定的输入栅格文件中创建一个VRT数据。从gdal1.6开始支持。

该工具没有进行说明。

### 4. gdal\_contour DEM生成等高线

用法：

gdal\_contour [-b <band>] [-a <attribute\_name>] [-3d] [-inodata]

                    [-snodata n] [-f <formatname>] [-i <interval>]

                    [-off <offset>] [-fl <level> <level>...]

                    [-nln <outlayername>]

                    <src\_filename> <dst\_filename>

参数说明：

使用DEM生成等高线矢量数据。

-b band:

指定要处理的波段序号，默认为第一个波段。

-a name:

指定一个属性表的字段名称用于保存高程值，如果不指定则不会在属性表中填写。

-3d:

强制使用3D定点而不是2D，即高程值将被写入等高线矢量中。

-inodata:

忽略任何nodata值。

-snodata value:

设置输入文件的nodata值。

-f format:

创建输出矢量的格式，默认为ESRI的shape文件。

-i interval:

等高线间距。

-off offset:

设置高程相对系数。

-fl level:

Name one or more "fixed levels" to extract.

-nln outlayername:

指定输出矢量文件的图层名称，默认为“contour”。

举例：

使用dem.tif生成一个等高距为10米的等高线矢量文件contour.shp。

gdal\_contour -a elev dem.tif contour.shp -i 10.0

### 5. gdaldem DEM算法

用法：

- To generate a shaded relief map from any GDAL-supported elevation raster :

    gdaldem hillshade input\_dem output\_hillshade

                [-z ZFactor (default=1)] [-s scale\* (default=1)]"

                [-az Azimuth (default=315)] [-alt Altitude (default=45)]

                [-alg ZevenbergenThorne]

                [-compute\_edges] [-b Band (default=1)] [-of format] [-co "NAME=VALUE"]\* [-q]

- To generate a slope map from any GDAL-supported elevation raster :

    gdaldem slope input\_dem output\_slope\_map"

                [-p use percent slope (default=degrees)] [-s scale\* (default=1)]

                [-alg ZevenbergenThorne]

                [-compute\_edges] [-b Band (default=1)] [-of format] [-co "NAME=VALUE"]\* [-q]

- To generate an aspect map from any GDAL-supported elevation raster

  Outputs a 32-bit float raster with pixel values from 0-360 indicating azimuth :

    gdaldem aspect input\_dem output\_aspect\_map"

                [-trigonometric] [-zero\_for\_flat]

                [-alg ZevenbergenThorne]

                [-compute\_edges] [-b Band (default=1)] [-of format] [-co "NAME=VALUE"]\* [-q]

- To generate a color relief map from any GDAL-supported elevation raster

    gdaldem color-relief input\_dem color\_text\_file output\_color\_relief\_map

                [-alpha] [-exact\_color\_entry | -nearest\_color\_entry]

                [-b Band (default=1)] [-of format] [-co "NAME=VALUE"]\* [-q]

    where color\_text\_file contains lines of the format "elevation\_value red green blue"

- To generate a Terrain Ruggedness Index (TRI) map from any GDAL-supported elevation raster:

    gdaldem TRI input\_dem output\_TRI\_map

                [-compute\_edges] [-b Band (default=1)] [-of format] [-q]

- To generate a Topographic Position Index (TPI) map from any GDAL-supported elevation raster:

    gdaldem TPI input\_dem output\_TPI\_map

                [-compute\_edges] [-b Band (default=1)] [-of format] [-q]

- To generate a roughness map from any GDAL-supported elevation raster:

    gdaldem roughness input\_dem output\_roughness\_map

                [-compute\_edges] [-b Band (default=1)] [-of format] [-q]

参数说明：

该工具提供了7个模块。分别如下：

[hillshade](http://blog.csdn.net/liminlu0314/article/details/7218543#gdaldem_hillshade)（山地阴影）

用来生成山地阴影图像。

[slope](http://blog.csdn.net/liminlu0314/article/details/7218543#gdaldem_slope)

用来生成坡度图像。

[aspect](http://blog.csdn.net/liminlu0314/article/details/7218543#gdaldem_aspect)

用来生成坡向图像。

[color-relief](http://blog.csdn.net/liminlu0314/article/details/7218543#gdaldem_color_relief)

用来生成一个颜色渲染地图。

[TRI](http://blog.csdn.net/liminlu0314/article/details/7218543#gdaldem_TRI)

用来生成地表耐用指数（TRI）图像。

[TPI](http://blog.csdn.net/liminlu0314/article/details/7218543#gdaldem_TPI)

用来生成地形位置指数（TPI）图像。

[roughness](http://blog.csdn.net/liminlu0314/article/details/7218543#gdaldem_roughness)

用来生成粗糙度图像。

下面这些选项是通用选项：

input\_dem:

输入DEM数据。

output\_xxx\_map:

输出处理的结果数据。

-of format:

输出文件格式，默认为GeoTiff。

-compute\_edges:

(GDAL >= 1.8.0) 处理栅格边界数据和nodata附近的数据。

-alg ZevenbergenThorne:

(GDAL >= 1.8.0) 使用Zevenbergen & Thorne 规则代替Horn规则计算坡度坡向。研究表明Zevenbergen & Thorne用于平缓的地形，而Horn在处理比较粗糙的地形比较好。

-b band:

选择要处理的波段序号，默认为1.

-co "NAME=VALUE":

创建输出文件选项，根据创建的文件格式不同而不同。

-q:

安静模式，不输出进度信息。

在所有的算法中，除了颜色渲染，其他的算法全部是基于一个3x3的窗口运算，如果在这个窗口中有一个值为nodata，那么结果值就是nodata，在GDAL1.8之后，如果指定了-compute\_edges那么会在遇到nodata值时，会使用插值方式来确定nodata值，然后再进行处理。

算法模块

山地阴影

这个模块输出是一个8位栅格图，他在地形可视化中非常有用。您可以指定太阳高度角和太阳方位角海拔高度，垂直夸张因子和比例因子来改变结果。0作为输出的nodata值，具体的其他参数如下：

-z zFactor:

垂直夸张系数。

-s scale:

垂直系数的单位，如果水平单位是度，比如WGS84投影下的经纬度，可以将该参数设置为111120那么垂直单位是米，如果设置为370400，那么垂直单位为英尺。

-az azimuth:

光线方位角，度单位，0为正上方，90度为东边（右边）以此类推，默认值为315度。

-alt altitude:

光线的高度角，度单位，90度为DEM的正上方，0度为水平。

坡度

此模块将DEM栅格计算输出坡度值是32位浮数的栅格图像。你必须指定你想要的坡度值类型的选项：度或百分比坡度。同时也可以提供一个比例因子。-9999的值是用来作为输出NODATA值。以下具体的选项可供选择：

-p :

如果指定该选项，将使用坡度百分比计算，其他情况下使用度单位计算。

-s scale:

垂直系数的单位，如果水平单位是度，比如WGS84投影下的经纬度，可以将该参数设置为111120那么垂直单位是米，如果设置为370400，那么垂直单位为英尺。

坡向

该模块输出为一个32位浮点数的栅格数据，其值范围为0度到360度之间的坡向图， 0°表示坡向超正北方， 90°坡向朝向正东方，180°坡向朝向正南方，270°坡向朝向正西方。-9999表示无效值包括平地即坡度等于0的地方，详细的参数说明如下：

-trigonometric:

返回三角角度而不是方位，0°是指北，90°东，180°西，270°南。

-zero\_for\_flat:

返回0表示平地，而不是-9999。

如果使用这两个选项，使用gdaldem工具生成的坡度坡向结果同GRASS中的r.slope.aspect工具生成的结果一致。否则结果和Matthew Perry的aspect.cpp工具计算的结果一致。

color-relief

该功能是使用DEM数据和一个文本存储的颜色配置文件计算输出一个三波段（RGB）或者四波段（RGBA）的栅格图像。颜色配置文件中存储的是高程值和其对应的颜色值。默认情况下，给定的两个高程值之间的颜色使用该高程值对应的颜色插值来进行计算。参数-exact\_color\_entry或者参数-nearest\_color\_entry可以避免使用线性插值来确定从配置文件中没有找到对应的颜色值。

同时还有下面的选项可以使用：

color\_text\_file:

基于文本的颜色配置文件

-alpha :

在输出文件中添加Alpha通道

-exact\_color\_entry :

使用严格的匹配，在配置文件中搜索对应的高程值，如果没有找到匹配的颜色，使用“0,0,0,0”来进行填充。

-nearest\_color\_entry :

在颜色配置文件中使用与RGBA颜色相近的颜色进行填充。

color-relief是支持输出VRT格式的，在这种情形下，会将颜色配置文件转换为一个LUT（Look up table颜色查找表）。值得注意的是，使用百分比指定的高程值会被转换为绝对高程值。

基于文本格式的颜色配置文件一般每行都有四列，分别是：高程值，红，绿，蓝（0-255之间）。高程值可以是任何浮点数，或者使用nv来表示无效值。高程值统一可以使用百分比来表示，0%表示DEM中的最小值，100%表示DEM数据中的最大值。

此外还有一列是可选的就是Alpha部分，如果没有指定，默认值为255，表示不透明。同时每列之间的分隔符可以是：逗号（，），制表符（tab键），空格（ ），冒号（:），以上符合均为英文输入法下的符号。通常使用的颜色单词也可以用来替换RGB值，包括： white白色, black黑色, red红色, green绿色, blue蓝色, yellow黄色, magenta品红, cyan青色, aqua浅绿色, grey/gray灰色, orange橙色, brown棕色, purple/violet 紫色和indigo靛蓝色。

从GDAL1.8.0开始，GMT的.cpt调色板文件页可以支持（仅在COLOR\_MODEL = RGB时可用）。

注意：颜色配置文件的句法同GRASS中的r.colors工具一致，ESRI的HDR颜色表文件（.clr）也是这种格式，Alpha部分的支持是GDAL的一个特殊扩展。

颜色配置文件示例：

3500   white

2500   235:220:175

50%   190 185 135

700    240 250 150

0      50  180  50

nv     0   0   0   0

TRI

该工具生成一个单波段的栅格图像，TRI是地形粗糙指数（Terrain Ruggedness Index）的缩写，代表的是中心点象元与周围8个象元点的高程差的绝对值的平均值。(参考Wilson et al 2007, Marine Geodesy 30:3-35)。

输出文件中的-9999代表无效值。该功能没有其他选项。

TPI

该工具生成一个单波段的栅格图像，TPI是地形方位指数（Topographic Position Index）的缩写，代表的是中心点象元高程与周围8个象元点的高程平均值的差值。(参考Wilson et al 2007, Marine Geodesy 30:3-35)。

输出文件中的-9999代表无效值。该功能没有其他选项。

roughness

该工具使用DEM数据计算一个单波段的栅格图像， 粗糙度是指当前窗口中的最大值和最小值的差值。(参考Wilson et al 2007, Marine Geodesy 30:3-35)。

输出文件中的-9999代表无效值。该功能没有其他选项。

### 6. gdal\_rasterize 矢量栅格化

用法：

gdal\_rasterize [-b band]\* [-i] [-at]

       [-burn value]\* | [-a attribute\_name] [-3d]

       [-l layername]\* [-where expression] [-sql select\_statement]

       [-of format] [-a\_srs srs\_def] [-co "NAME=VALUE"]\*

       [-a\_nodata value] [-init value]\*

       [-te xmin ymin xmax ymax] [-tr xres yres] [-tap] [-ts width height]

       [-ot {Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/

             CInt16/CInt32/CFloat32/CFloat64}] [-q]

       <src\_datasource> <dst\_filename>

参数说明：

该程序将矢量几何图形（点，线，面）栅格化到栅格波段中。矢量必须是OGR库支持的矢量格式。注意的是矢量数据和栅格数据必须具有相同的坐标系统，不支持动态重投影功能。从GDAL1.8.0开始，支持创建结果图像，必须使用-tr或者-ts选项才行。

-b band:

要写入的栅格波段，可以使用-b参数来进行多选，默认是写入第一个波段中。

-i:

栅格反转，. Burn the fixed burn value, or the burn value associated with the first feature into all parts of the image not inside the provided a polygon.

-at:

Enables the ALL\_TOUCHED rasterization option so that all pixels touched by lines or polygons will be updated not just those one the line render path, or whose center point is within the polygon. Defaults to disabled for normal rendering rules.

-burn value:

栅格象元值，将所有的对象都用该值写入波段中，可以使用一个列表，来指定每个波段写入的象元值。

-a attribute\_name:

指定属性字段中的字段值作为栅格值写入栅格文件中。

-3d:

将坐标点的Z值写入栅格中，假如参数-burn和-a attribute\_name 被指定，那么该参数将会无效，目前仅仅支持3D的点和线数据。

-l layername:

指定矢量文件中的图层名作为输入要素，可以指定多次，但是至少有一个图层名或者-sql必须被指定。

-where expression:

SQL WHERE格式的查询语句，从指定的图层中选择特殊的要素。

-sql select\_statement:

使用SQL语句来指定矢量数据中那些要素将要被栅格化。

-of format:

(GDAL >= 1.8.0)输出文件格式，默认为GeoTiff格式，使用短格式名称。

-a\_nodata value:

(GDAL >= 1.8.0) 指定输出文件的无效值。

-init value:

(GDAL >= 1.8.0) 指定初始化栅格波段中的象元值，但是该值不能作为输出文件的无效值，如果只指定了一个值，那么其他的波段都会使用该值来填充。

-a\_srs srs\_def:

(GDAL >= 1.8.0) 覆盖输出文件的投影，如果没有指定，将使用输入的矢量文件中的投影作为输出投影，如果输入和输出投影不一致，将尝试重投影矢量。该参数的格式可以是GDAL/OGR支持的各种投影格式，如完整的WKT，PROJ.4，EPSG:n或者存储WKT的文本文件。

-co "NAME=VALUE":

(GDAL >= 1.8.0) 创建图像选项，具体参考不同的栅格格式。

-te xmin ymin xmax ymax :

(GDAL >= 1.8.0) 设置地理范围，该值必须使用地理单位，如果没有指定，该范围同输入的矢量图层范围一致。

-tr xres yres :

(GDAL >= 1.8.0)设置输出分辨率，该值必须是有地理单位，两个必须都是正数。

-tap:

(GDAL >= 1.8.0) (target aligned pixels) align the coordinates of the extent of the output file to the values of the -tr, such that the aligned extent includes the minimum extent.（不清楚该参数，同gdalwarp中的-tap）

-ts width height:

(GDAL >= 1.8.0) 设置输出文件大小，注意使用-ts是不能同时使用-tr。

-ot type:

(GDAL >= 1.8.0) 输出波段的数据类型，默认为64位浮点数。

-q:

(GDAL >= 1.8.0) 不输出进度信息和错误信息。

src\_datasource:

任何OGR支持的矢量数据。

dst\_filename:

GDAL支持的输出文件，必须支持更新模式访问，在GDAL1.8.0之前，该工具不能创建结果图像。

举例：

下面的例子，将mask.shp中的所有多边形栅格化为一个RGB的Tif图像work.tif，使用红色表示（RGB = 255,0,0）。

gdal\_rasterize -b 1 -b 2 -b 3 -burn 255 -burn 0 -burn 0 -l mask mask.shp work.tif

下面的例子将所有的class属性值为A的全部要素输出到一个高程文件中，高程文件的高程值为属性字段ROOF\_H 的属性值。

gdal\_rasterize -a ROOF\_H -where 'class="A"' -l footprints footprints.shp city\_dem.tif

### 7. gdaltransform 坐标系转换工具

用法：

gdaltransform [--help-general]

    [-i] [-s\_srs srs\_def] [-t\_srs srs\_def] [-to "NAME=VALUE"]

    [-order n] [-tps] [-rpc] [-geoloc]

    [-gcp pixel line easting northing [elevation]]\*

    [srcfile [dstfile]]

参数说明：

gdaltransform工具用来转换坐标，从支持的投影，包括GCP点的变换。

-s\_srs srs def:

原始空间参考，该值将被OGRSpatialReference.SetFromUserInput()函数调用，支持包括EPSG PCS 和GCSes (如EPSG:4296), PROJ.4声明或者一个后缀名为.prf的文件存储的WKT串。

-t\_srs srs\_def:

目标空间参考，该值将被OGRSpatialReference.SetFromUserInput()函数调用，支持包括EPSG PCS 和GCSes (如EPSG:4296), PROJ.4声明或者一个后缀名为.prf的文件存储的WKT串。

-to NAME=VALUE:

设置转换参数，将会使函数[GDALCreateGenImgProjTransformer2()](http://blog.csdn.net/liminlu0314/article/details/7218543#a94cd172f78dbc41d6f407d662914f2e3)来调用。

-order n:

几何多项式变换次数（1到3），默认情况下会根据输入的GCP点的个数自动确定。

-tps:

强制将GCP点使用TPS转换方式转换。

-rpc:

强制使用RPC参数转换。

-geoloc:

强制使用Geolocation数组转换。

-i

逆转换，从目标到原始投影转换。

-gcppixel line easting northing [elevation]:

指定GCP点，通常至少需要三个。

srcfile:

原始投影下定义的GCP点文件，如果没有指定，需要从命令行-s\_srs或者-gcp参数输入。

dstfile:

目标投影定义文件。

读取的坐标必须是成对出现（或者三个一组），输出也以同样的方式输出，所有的变换共gdalwarp相同，包括使用GCP的变换。注意输入输出必须使用十进制小数，当前不支持DMS（度分秒）的输入输出。如果指定了输入图像，那么输入的坐标是基于图像的行列号，如果输出图像指定，输出的坐标也是图像的行列号。

举例：

重投影示例

简单的投影转换，从一个投影转换到另一个投影：

gdaltransform -s\_srs EPSG:28992 -t\_srs EPSG:31370

177502 311865

Produces the following output in meters in the "Belge 1972 / Belgian Lambert 72" projection:

244510.77404604 166154.532871342 -1046.79270555763

图像RPC变换示例：

下面的命令行使用基于RPC变换，并且使用-i选项来将经纬度坐标转为图像的行列号：

gdaltransform -i -rpc 06OCT20025052-P2AS-005553965230\_01\_P001.TIF

125.67206 39.85307 50

Produces this output measured in pixels and lines on the image:

3499.49282422381 2910.83892848414 50

### 8. nearblack

用法：

gdalinfo [--help-general] [-mm] [-stats] [-hist] [-nogcp] [-nomd]

         [-noct] [-nofl] [-checksum] [-proj4] [-mdd domain]\*

 [-sd subdataset] datasetname

参数说明：

该工具没有进行说明。

### 9. gdal\_grid 插值工具

用法：

gdal\_grid [-ot {Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/

          CInt16/CInt32/CFloat32/CFloat64}]

  [-of format] [-co "NAME=VALUE"]

  [-zfield field\_name]

  [-a\_srs srs\_def] [-spat xmin ymin xmax ymax]

          [-clipsrc <xmin ymin xmax ymax>|WKT|datasource|spat\_extent]

          [-clipsrcsql sql\_statement] [-clipsrclayer layer]

          [-clipsrcwhere expression]

  [-l layername]\* [-where expression] [-sql select\_statement]

  [-txe xmin xmax] [-tye ymin ymax] [-outsize xsize ysize]

  [-a algorithm[:parameter1=value1]\*] [-q]

  <src\_datasource> <dst\_filename>

参数说明：

该程序从OGR支持的矢量数据中创建一个规则的栅格数据。输入数据必须有用来插值的数据，可以选择下面的插值方法。

-ot type:

指定输出文件个数据类型。

-of format:

输出文件格式，默认为GeoTiff格式。

-txe xmin xmax:

设置输出文件的地理范围X。

-tye ymin ymax:

设置输出文件的地理范围Y。

-outsize xsize ysize:

设置输出文件的大小。

-a\_srs srs\_def:

设置输出文件的投影，可以是WKT串，PROJ.4，EPSG:n或者含有wkt的文件。

-zfield field\_name:

指定矢量要素中获取Z值的字段。

-a [algorithm[:parameter1=value1][:parameter2=value2]...]:

设置插值算法或者数据单位参数，具体参考 [INTERPOLATION ALGORITHMS](http://blog.csdn.net/liminlu0314/article/details/7218543#gdal_grid_algorithms) 和[DATA METRICS](http://blog.csdn.net/liminlu0314/article/details/7218543#gdal_grid_metrics) 。

-spat xmin ymin xmax ymax:

指定空间范围进行过滤(xmin, ymin) - (xmax, ymax)。

-clipsrc [xmin ymin xmax ymax]|WKT|datasource|spat\_extent:

添加一个空间过滤来选择处理的要素，包括空间范围（SRS表示），WKT几何形状（POLYGON 或MULTIPOLYGON），也可以使用下面的参数来进行过滤：-clipsrclayer，-clipsrcwhere或-clipsrcsql。

-clipsrcsql sql\_statement:

使用SQL语句对原始数据进行查询过滤。

-clipsrclayer layername:

指定源数据中图层名称。

-clipsrcwhere expression:

使用属性表来限制几何体。

-l layername:

指定图层名称。

-where expression:

使用SQL WHERE方式来过滤图层中的数据。

-sql select\_statement:

使用SQL语句来过滤图层中的数据。

-co "NAME=VALUE":

创建文件参数选项。

-q:

不输出进度信息和错误信息。

src\_datasource:

输入矢量，任何OGR库支持的矢量数据。

dst\_filename:

输出栅格，GDAL支持的栅格数据。

INTERPOLATION ALGORITHMS（插值算法）

插值算法可以选择下面几种：

invdist

距离次方反比法，这个是默认算法，该算法还有下列参数：

power:

权重指数，默认为2.0。

smoothing:

平滑参数，默认为0.0。

radius1:

搜索椭圆第一半径（X轴的旋转角度为0），设置该参数为0时将使用所有的点数组，默认为0.0。

radius2:

搜索椭圆第二半径（Y轴的旋转角度为0），设置该参数为0时将使用所有的点数组，默认为0.0。

angle:

搜索椭圆的旋转角度，使用度单位。默认为0.0，逆时针方向。

max\_points:

使用的数据点的最大个数，不会搜索多余这个数字的点，如果搜索椭圆设置将会使用该值（两个半径均不为0时）。默认为0，为0表示使用所有的点。

min\_points:

使用的最小数据点个数，如果数据点小于该值，并且找到的网络节点值为空，将使用Nodata值来标记。该值仅仅当搜索椭圆设置（即两个半径均不为0），默认该值为0。

nodata:

设置NODATA值，默认为0.0。

average

移动均值算法，该算法参数如下：

radius1:

搜索椭圆第一半径（X轴的旋转角度为0），设置该参数为0时将使用所有的点数组，默认为0.0。

radius2:

搜索椭圆第二半径（Y轴的旋转角度为0），设置该参数为0时将使用所有的点数组，默认为0.0。

angle:

搜索椭圆的旋转角度，使用度单位。默认为0.0，逆时针方向。

min\_points:

使用数据点的最小个数，如果数据点小于该值，并且找到的网络节点值为空，将使用Nodata值来标记，默认该值为0。

nodata:

设置NODATA值。默认值为0.0。

注意：使用移动均值算法必须设置搜索椭圆，当计算网格节点值时使用窗口的均值。

nearest

最邻近算法。该算法参数如下：

radius1:

搜索椭圆第一半径（X轴的旋转角度为0），设置该参数为0时将使用所有的点数组，默认为0.0。

radius2:

搜索椭圆第二半径（Y轴的旋转角度为0），设置该参数为0时将使用所有的点数组，默认为0.0。

angle:

搜索椭圆的旋转角度，使用度单位。默认为0.0，逆时针方向。

nodata:

设置NODATA值。默认值为0.0。

DATA METRICS（数据单位）

Besides the interpolation functionality [gdal\_grid](http://liml/gdal/html/gdal_grid.html) 除了插值功能外，还有使用指定窗口和输出网格来计算一些数据指标，这些指标有：

minimum:

搜索椭圆中网格节点的最小值。

maximum:

搜索椭圆中网格节点的最大值。

range:

在搜索椭圆中查找到的最大值和最小值的差值。

count:

在搜索椭圆中找到的点个数。

average\_distance:

网格节点（搜索椭圆中心）和所有在网格节点搜索椭圆的数据点之间的平均距离。

average\_distance\_pts:

在网格节点搜索椭圆中发现的数据点之间的平均距离。椭圆内每对点之间距离的计算和所有的距离平均是作为一个网格节点的值设置。

所有的指标都具有下面相同的选项：

radius1:

搜索椭圆第一半径（X轴的旋转角度为0），设置该参数为0时将使用所有的点数组，默认为0.0。

radius2:

搜索椭圆第二半径（Y轴的旋转角度为0），设置该参数为0时将使用所有的点数组，默认为0.0。

angle:

搜索椭圆的旋转角度，使用度单位。默认为0.0，逆时针方向。

min\_points:

使用的数据点的最小个数。如果找到的数据点小于将设置为NODATA值，该值仅仅用于设置搜索椭圆时可用（两个半径均不为0时）。默认值为0。

nodata:

设置NODATA值。默认值为0.0。

READING COMMA SEPARATED VALUES（读取逗号分割符文件CSV）

通常使用一个文本文件，使用逗号分隔XYZ值，该文件通常叫做CSV文件。可以在[gdal\_grid](http://liml/gdal/html/gdal_grid.html)中使用该文件。你需要做的就是要对你的CSV文件创建一个VRT文件头，并将该文件作为[gdal\_grid](http://liml/gdal/html/gdal_grid.html)的输入数据。关于VRT格式详情参考 [Virtual Format](http://liml/gdal/html/ogr/drv_vrt.html) 页面。

下面有个小例子。假如我们有个CSV文件叫dem.csv ，里面内容为：

Easting,Northing,Elevation

86943.4,891957,139.13

87124.3,892075,135.01

86962.4,892321,182.04

87077.6,891995,135.01

...

对于上面的数据，创建一个dem.vrt 的头文件，内容如下：

<OGRVRTDataSource>

    <OGRVRTLayer name="dem">

        <SrcDataSource>dem.csv</SrcDataSource>

<GeometryType>wkbPoint</GeometryType>

<GeometryField encoding="PointFromColumns" x="Easting" y="Northing" z="Elevation"/>

    </OGRVRTLayer>

</OGRVRTDataSource>

该说明是一个叫做2.5D的几何数据，含有三个坐标XYZ，Z值将被用来插值。现在可以使用dem.vrt来进行测试，可以先使用[ogrinfo](http://liml/gdal/html/ogrinfo.html)来测试该文件是否可用。数据中包括一个叫dem的点图层，数据包括在CSV文件中。使用这种方式，可以处理超过三列，交换列的CSV文件。

如果你的CSV文件不包括列头，那么可以在VRT头文件中加入下面的内容来进行处理：<GeometryField encoding="PointFromColumns" x="field\_1" y="field\_2" z="field\_3"/>

[Comma Separated Value](http://liml/gdal/html/ogr/drv_csv.html) 页面描述了OGR库对CSV文件的支持描述信息。

举例：

下面将创建一个TIFF的图像数据，使用VRT数据源，从一个CSV文件中，使用反距离次方算法来进行插值。

gdal\_grid -a invdist:power=2.0:smoothing=1.0 -txe 85000 89000 -tye 894000 890000 -outsize 400 400 -of GTiff -ot Float64 -l dem dem.vrt dem.tiff

下面的命令行同上面的类似，但是读取的数据是使用-zfield 选项来指定的。

gdal\_grid -zfield "Elevation" -a invdist:power=2.0:smoothing=1.0 -txe 85000 89000 -tye 894000 890000 -outsize 400 400 -of GTiff -ot Float64 -l dem dem.vrt dem.tiff

### 10. gdallocationinfo 栅格查询工具

用法：

gdallocationinfo [--help-general] [-xml] [-lifonly] [-valonly]

               [-b band]\* [-l\_srs srs\_def] [-geoloc] [-wgs84]

               srcfile [x y]

参数说明：

Gdallocationinfo程序通过指定一种坐标系中的坐标来进行对其位置的像元值进行输出显示。

-xml:

输出XML格式，方便后期处理。

-lifonly:

输出LocationInfo 中的信息。

-valonly:

仅仅输出指定位置的每个波段的像元值。

-b band:

选择一个波段进行处理，多波段文件将被列出显示。默认是查询所有的波段。

-l\_srs srs def:

指定输入的x,y坐标的坐标系。

-geoloc:

表示输入的x,y坐标是地理参考坐标系。

-wgs84:

表示输入的x,y是WGS84坐标系下的经纬度坐标。

srcfile:

输入栅格图像的名称。

x:

查询的X坐标。默认为图像行列号，如果使用-l\_srs,-wgs84或者-geoloc时按照指定的坐标来处理。

y:

查询的Y坐标。默认为图像行列号，如果使用-l\_srs,-wgs84或者-geoloc时按照指定的坐标来处理

该工具的目的是输出一个像素的各种信息。目前支持下面四项：

· 像素的行列号。

· 输出元数据中的LocationInfo 信息，目前只支持VRT文件。

· 全部波段或子文件中的所有波段中的像元值。

· 未缩放的像元值，如果对波段进行了缩放和偏移操作。

输入x和y坐标是，是通过命令行来进行输入的，一般是先x后y，即先列号后行号；如果使用-geoloc, -wgs84, 或-l\_srs 选项后，会自动根据输入进行交换。默认的输出信息是纯文本文件，也可以使用-xml选项将其使用xml格式进行输出。将来会添加其他的信息。

### 11. gdalsrsinfo 将给定的CRS按照不同的格式显示

用法：

gdalsrsinfo [options] srs\_def

列出给定的CRS的各种表示方法。如ESRI的wkt格式，OGC的wkt格式和Proj4的格式等。

srs\_def 可以是一个GDAL或者OGR支持的文件，或者是任何GDAL和OGR支持的CRS格式（包括WKT,PROJ4,EPSG:n等）。

参数说明：

Options:

   [--help-general] [-h]   显示帮助信息并退出

   [-p]                输出信息（如WKT字串格式）

   [-V]                验证SRS

   [-o out\_type]        输出类型，默认为all，所有类型见下：

                      {all,proj4,wkt,wkt\_simple,wkt\_old,wkt\_esri,mapinfo,xml}

举例：

gdalsrsinfo  -o all  "EPSG:4326"

PROJ.4 : '+proj=longlat +datum=WGS84 +no\_defs '

OGC WKT :

GEOGCS["WGS 84",

    DATUM["WGS\_1984",

        SPHEROID["WGS 84",6378137,298.257223563,

            AUTHORITY["EPSG","7030"]],

        AUTHORITY["EPSG","6326"]],

    PRIMEM["Greenwich",0,

        AUTHORITY["EPSG","8901"]],

    UNIT["degree",0.0174532925199433,

        AUTHORITY["EPSG","9122"]],

    AUTHORITY["EPSG","4326"]]

OGC WKT (simple) :

GEOGCS["WGS 84",

    DATUM["WGS\_1984",

        SPHEROID["WGS 84",6378137,298.257223563]],

    PRIMEM["Greenwich",0],

    UNIT["degree",0.0174532925199433]]

ESRI WKT :

GEOGCS["GCS\_WGS\_1984",

    DATUM["D\_WGS\_1984",

        SPHEROID["WGS\_1984",6378137,298.257223563]],

    PRIMEM["Greenwich",0],

    UNIT["Degree",0.017453292519943295]]

### 12. gdal-config GDAL配置参数

用法：

gdal-config [OPTIONS]

Options:

        [--prefix[=DIR]]

        [--libs]

        [--cflags]

        [--version]

        [--ogr-enabled]

        [--formats]

参数说明：

该脚本工具可以用来查看关于GDAL的一些安装信息等内容。

--prefix:

显示GDAL的安装目录。

--libs:

显示GDAL所依赖的库。

--cflags:

编译GDAL时包括的宏定义等信息。

--version:

显示GDAL版本信息。

--ogr-enabled:

显示是否包含OGR库，如果输出"yes"说明OGR可以使用，如果是"no"则不能使用OGR库中的东西。

--formats:

输出GDAL支持的格式。

[GDAL源码剖析（五）之Python命令行程序](http://blog.csdn.net/liminlu0314/article/details/7218552)

http://blog.csdn.net/liminlu0314/article/details/7218552

# 一、 GDAL Python工具

本文主要介绍的是GDAL工具集中的Python脚本命令，需要的环境必须是有Python环境和GDAL的Python版本。这是必须的，否则这些工具都不能用。对于已经安装ArcGIS的同学来说，Python都已经安装好了，可以直接下载GDAL的Python版本，然后就可以使用下面这些工具。

### 1. rgb2pct.py 转换24位RGB图为8位图

用法：

rgb2pct.py [-n colors | -pct palette\_file] [-of format] source\_file dest\_file

参数说明：

该工具会自动根据指定的RGB图像计算最合适的假彩色颜色表。然后对结果影像使用该颜色表。简而言之，就是将RGB彩色图像转换为一个单波段的图像，使用颜色表来表示颜色。

-n colors:

指定生成颜色表的颜色数目，默认是256，其值必须是2到256之间的整数值。

-pct palette\_file:

从调色板文件中提取颜色表而不从图像中计算。调色板文件必须是GDAL支持的调色板格式。

-of format:

输出文件格式，默认为GeoTiff格式，而且输出格式必须支持颜色表。

source\_file:

输入的RGB图像。

dest\_file:

输出的图像路径。如果图像不存在会创建一个。

举例：

如果希望指定调色板信息，比如简单的文本格式，如GDAL VRT格式，在下面的例子中将指定一个使用文本编辑器创建的VRT格式的调色板文件，一共有四个颜色，RGBA值分别是：238/238/238/255，237/237/237/255，36/236/236/255和229/229/229/255。

% rgb2pct.py -pct palette.vrt rgb.tif pseudo-colored.tif

% more < palette.vrt

<VRTDataset rasterXSize="226" rasterYSize="271">

  <VRTRasterBand dataType="Byte" band="1">

    <ColorInterp>Palette</ColorInterp>

    <ColorTable>

      <Entry c1="238" c2="238" c3="238" c4="255"/>

      <Entry c1="237" c2="237" c3="237" c4="255"/>

      <Entry c1="236" c2="236" c3="236" c4="255"/>

      <Entry c1="229" c2="229" c3="229" c4="255"/>

    </ColorTable>

  </VRTRasterBand>

</VRTDataset>

### 2. pct2rgb.py 转换8位图为24位RGB图

用法：

pct2rgb.py [-of format] [-b band] [-rgba] source\_file dest\_file

参数说明：

该工具是将一个带有颜色表的图像转为RGB图像。

-of format:

输出文件格式，默认为GeoTiff格式。

-b band:

指定要转换的波段序号，默认是第一个波段。

-rgba:

生成RGBA文件（默认是生成RGB文件）。

source\_file:

输入文件。

dest\_file:

输出文件。

### 3. gdal\_merge.py 镶嵌图像

用法：

gdal\_merge.py [-o out\_filename] [-of out\_format] [-co NAME=VALUE]\*

              [-ps pixelsize\_x pixelsize\_y] [-tap] [-separate] [-v] [-pct]

              [-ul\_lr ulx uly lrx lry] [-n nodata\_value] [-init "value [value...]"]

              [-ot datatype] [-createonly] input\_files

参数说明：

该工具会自动镶嵌指定的图像。所有的图像必须具有相同的坐标系统和相同的波段数目；输入图像可能是有重叠区或者不同的分辨率。在重叠区部分最后的图像将会覆盖之前的图像值。

-o out\_filename:

输出文件，如果不存在将会创建一个新图像。如果该值不指定，将会创建一个叫out.tif的图像。

-of format:

输出文件格式，默认为GeoTiff格式。

-co NAME=VALUE:

创建图像选项，具体参考具体图像格式说明。

-ot datatype:

指定输出数据类型，使用类型名称，如Byte，Int16等。

-ps pixelsize\_x pixelsize\_y:

输出图像的象元大小，如果不指定，将以第一个图像的分辨率为基准。

-tap:

(GDAL >= 1.8.0) (target aligned pixels) align the coordinates of the extent of the output file to the values of the -tr, such that the aligned extent includes the minimum extent.

-ul\_lr ulx uly lrx lry:

输出文件范围，如果不指定，将会以所有的输入图像的范围并集为输出文件范围。

-v:

生成详细的镶嵌操作信息，当执行结束的时候。

-separate:

Place each input file into a separate stacked band.

-pct:

提取颜色表从第一个输入图像，并在结果图像中使用，按照这种方式镶嵌图像的话，后面的图像的颜色表都会被忽略，以第一个为准。

-n nodata\_value:

指定nodata值，在镶嵌操作的时候会忽略该值。

-init "value(s)":

指定输出文件的初始值，不会在输出文件中指定nodata值，如果只指定了一个值，那么所有的波段都会使用该值来作为初始值。

-createonly:

输出文件已经创建好了。但是输入文件数据没有写入。

举例：

创建一个图像，将所有的波段都指定为255。

% gdal\_merge.py -init 255 -o out.tif in1.tif in2.tif

创建一个RGB图像，并将前两个波段初始化为0，第三个波段为255。

% gdal\_merge.py -init "0 0 255" -o out.tif in1.tif in2.tif

### 4. gdal2tiles.py 生成TMS切片

用法：

gdal2tiles.py [-title "Title"] [-publishurl http://yourserver/dir/]

              [-nogooglemaps] [-noopenlayers] [-nokml]

              [-googlemapskey KEY] [-forcekml] [-v]

              input\_file [output\_dir]

参数说明：

该工具按照OSGeo的切片地图服务说明书来生成小的切片数据和元数据。基于Google地图和OpenLayers简单的网页地图浏览。

GDAL2Tiles 工具创建Google Earth (KML SuperOverlay)必需的元数据文件，按照EPSG:4326投影。同时会创建世界文件（World file），但是页可以创建其他的投影的文件。

-title "Title":

生成元数据，网页视图和KML文件。

-publishurl <http://yourserver/dir/>:

添加发布的网页地址，会将结果上传到该地址。

-nogooglemaps:

不要生成基于Google地图的html页面。

-noopenlayers:

不要生成基于OpenLayers的html页面。

-nokml:

不要生成Google Earth的kml文件。

Do not generate KML files for Google Earth.

-googlemapskey KEY:

指定生成区域的范围，使用Google Maps的API。参考(<http://www.google.com/apis/maps/signup.html>).

-forcekml

生成kml文件，输入图像必需是EPSG:4326的投影！

-v

处理结束后输出详细信息。

### 5. gdal\_retile.py

用法：

gdal\_retile.py [-v] [-co NAME=VALUE]\* [-of out\_format] [-ps pixelWidth pixelHeight]

               [-ot  {Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/

                      CInt16/CInt32/CFloat32/CFloat64}]'

               [ -tileIndex tileIndexName [-tileIndexField tileIndexFieldName]]

               [ -csv fileName [-csvDelim delimiter]]

               [-s\_srs srs\_def]  [-pyramidOnly]

               [-r {near/bilinear/cubic/cubicspline/lanczos}]

               -levels numberoflevels

               [-useDirForEachRow]

               -targetDir TileDirectory input\_files

参数说明：

该工具对输入图像创建分块。所有的输入图像必需具有相同的投影和相同的波段数。可以选择生成金字塔。会生成输出瓦片的矢量文件。

-targetDir directory:

输出瓦片存放的目录。金字塔文件会存放在子文件夹中，子文件夹的命名从1开始，数字代表金字塔级别。创建的输出名称由源文件名称和一个序号组成。

-of format:

输出文件格式，默认为GeoTiff格式。

-co NAME=VALUE:

创建选项，具体参考具体的格式说明。

-ot datatype:

指定输出图像的数据类型，使用类型名称，如Byte，Int16，UInt16等。

-ps pixelsize\_x pixelsize\_y:

输出文件大小，如果不指定，默认是256\*256。

-levels numberOfLevels:

创建的金字塔级数。

-v:

输出相信的信息。

-pyramidOnly:

不建立瓦片文件，只建立金字塔。

-r algorithm:

重采样方式，默认为最邻近采样。

-s\_srs srs\_def:

源文件的空间参考。坐标系统可以是任何OGRSpatialReference.SetFro函数支持的坐标系统，包括EPSG PCS和GCSes，PROJ.4声明或者其他的包含WKT字符串的prf文件。如果不知道，将会从指定输入图像中读取。同时该空间参考用来创建生成的瓦片的矢量文件。

-tileIndex tileIndexName:

包含结果瓦片索引的shp名称。

-tileIndexField tileIndexFieldName:

包含瓦片名称的属性表字段名称。

-csv csvFileName:

指定包含瓦片地理信息的csv文件名称。该文件包含五列，分别是：（瓦片名称，最小x，最大x，最小y，最大y）tilename,minx,maxx,miny,maxy。

-csvDelim column delimiter:

CSV文件的列之间分隔符，默认使用分号风格。

-useDirForEachRow:

通常情况下，输出的瓦片文件存放在-targetDir指定的文件夹中。对于大图像，在有些文件系统中对于文件夹中的文件数过多可能会有一些问题，从而导致gdal\_retile程序不能正常执行完毕。使用该参数可以创建不同的目录结构。原始的瓦片文件存放在名字为0的子文件夹中，金字塔依次存放在子文件1，2，3...中，数字代表金字塔的级别。

### 6. gdal\_proximity.py

用法：

gdal\_proximity.py srcfile dstfile [-srcband n] [-dstband n]

                  [-of format] [-co name=value]\*

                  [-ot Byte/Int16/Int32/Float32/etc]

                  [-values n,n,n] [-distunits PIXEL/GEO]

                  [-maxdist n] [-nodata n] [-fixed-buf-val n]

参数说明：

gdal\_proximity.py脚本用来生成栅格图像的距离图，即计算中心点的象元与临近的象元之间的距离作为结果象元值。

srcfile

输入的栅格图像。=

dstfile

输出影像，可以是一个已经存在的文件，大小和原始图像一致，如果不存在，将会创建一个新图像。

-srcband n

指定用来计算的波段序号，默认为第一个波段。

-dstband n

指定输出的波段序号，默认为第一个波段。

-of format:

输出图像格式，默认为GeoTiff格式，使用短格式名称。

-co "NAME=VALUE":

创建图像选项，具体参考对应格式的说明。

-ot datatype:

输出文件的数据类型，使用短名称，如Byte，Int16。

-values n,n,n:

A list of target pixel values in the source image to be considered target pixels. If not specified, all non-zero pixels will be considered target pixels. 指定结果象元值在原始影像如果没有指定，所有的非0象元将被用来计算结果图。

-distunits PIXEL/GEO:

指定距离的单位，默认为像素单位，可以为像素单位或者地理坐标单位。

-maxdist n:

最大距离值，所有超过该值的象元具体将被设置为nodata值或者65535。距离使用像素单位时，如果使用GEO参数的话，不会执行该操作。

-nodata n:

指定nodata值。

-fixed-buf-val n:

Specify a value to be applied to all pixels that are within the -maxdist of target pixels (including the target pixels) instead of a distance value.

### 7. gdal\_polygonize.py 栅格矢量化工具

用法：

gdalinfo [--help-general] [-mm] [-stats] [-hist] [-nogcp] [-nomd]

         [-noct] [-nofl] [-checksum] [-proj4] [-mdd domain]\*

 [-sd subdataset] datasetname

参数说明：

gdalinfo程序输出gdal支持的栅格格式的一系列信息。

-mm

强制计算栅格每个波段的最大最小值。

-stats

读取和现实图像统计信息，如果指定该参数，将强制计算图像的统计信息，如各个波段的最大值、最小值、均值、标准差等。

举例：

Generate polygons from raster.

### 8. gdal\_sieve.py 小斑去除滤波工具

用法：

gdalinfo [--help-general] [-mm] [-stats] [-hist] [-nogcp] [-nomd]

         [-noct] [-nofl] [-checksum] [-proj4] [-mdd domain]\*

 [-sd subdataset] datasetname

参数说明：

gdalinfo程序输出gdal支持的栅格格式的一系列信息。

-mm

强制计算栅格每个波段的最大最小值。

-stats

读取和现实图像统计信息，如果指定该参数，将强制计算图像的统计信息，如各个波段的最大值、最小值、均值、标准差等。

举例：

Raster Sieve filter.

### 9. gdal\_fillnodata.py 填充NoData区域

用法：

gdalinfo [--help-general] [-mm] [-stats] [-hist] [-nogcp] [-nomd]

         [-noct] [-nofl] [-checksum] [-proj4] [-mdd domain]\*

 [-sd subdataset] datasetname

参数说明：

gdalinfo程序输出gdal支持的栅格格式的一系列信息。

-mm

强制计算栅格每个波段的最大最小值。

-stats

读取和现实图像统计信息，如果指定该参数，将强制计算图像的统计信息，如各个波段的最大值、最小值、均值、标准差等。

## 二、 其他说明

关于GDAL的Python编译、安装和使用方法，网上有很多的博文。由于Python是不用编译的脚本语言，所以不用编译就可以运行，语法简单，受到很多人的喜爱，我也是Python的爱好者。尤其是对于一些简单的功能，完全没有必要写一个C或者C++以及其他的语言程序，还需要编译后才能使用，总觉得不是那么的方便。如果用Python，写完后直接可以运行。

关于Python版本的编译和安装，参考此系列的博文三：<http://blog.csdn.net/liminlu0314/article/details/6945452>；此外，对于Python版本GDAL的使用，主要参考李林（lilin）写的《GDAL库学习笔记》系列文章（想当年我就是看这七篇文章来学习GDAL的）。链接地址为：<http://wiki.woodpecker.org.cn/moin/lilin>。

也可以直接下载编译好的Python版本GDAL，具体地址为：<http://pypi.python.org/pypi/GDAL/>

[GDAL源码剖析（六）之GDAL开发及其调试](http://blog.csdn.net/liminlu0314/article/details/7072007)

http://blog.csdn.net/liminlu0314/article/details/7072007