My Project

Contents

Chapter 1

GDAL Utilities

A collection of GDAL related programs.

The following utility programs are distributed with GDAL.

- gdalinfo - Report information about a file.
- gdal_translate - Copy a raster file, with control of output format.
- gdaladdo - Add overviews to a file.
- gdalwarp - Warp an image into a new coordinate system.
- gdaltindex - Build a MapServer raster tileindex.
- gdalbuildvrt - Build a VRT from a list of datasets.
- gdal_contour - Contours from DEM.
- gdaldem - Tools to analyze and visualize DEMs.
- rgb2pct.py - Convert a 24bit RGB image to 8bit paletted.
- pct2rgb.py - Convert an 8bit paletted image to 24bit RGB.
- gdal_merge.py - Build a quick mosaic from a set of images.
- gdal2tiles.py - Create a TMS tile structure, KML and simple web viewer.
- gdal_rasterize - Rasterize vectors into raster file.
- gdaltransform - Transform coordinates.
- nearblack - Convert nearly black/white borders to exact value.
- gdal_retile.py - Retiles a set of tiles and/or build tiled pyramid levels.
- gdal_grid - Create raster from the scattered data.
- gdal_proximity - Compute a raster proximity map.
- gdal_polygonize - Generate polygons from raster.
- gdal_sieve - Raster Sieve filter.
- gdal_fillnodata - Interpolate in nodata regions.
- gdallocationinfo - Query raster at a location.
- gdalsrsinfo - Report a given SRS in different formats. (GDAL >= 1.9.0)
- gdalmove - Transform the coordinate system of a file (GDAL >= 1.10)

- gdal_edit - Edit in place various information of an existing GDAL dataset (projection, geotransform, nodata, metadata)
- gdal_calc - Command line raster calculator with numpy syntax
- gdal_pansharpen.py - Perform a pansharpen operation.
- gdal-config - Get options required to build software using GDAL.
- gdalmanage - Identify, copy, rename and delete raster.
- gdalcompare - Compare two images and report on differences.

1.1 Creating New Files

Access an existing file to read it is generally quite simple. Just indicate the name of the file or dataset on the command line. However, creating a file is more complicated. It may be necessary to indicate the the format to create, various creation options affecting how it will be created and perhaps a coordinate system to be assigned. Many of these options are handled similarly by different GDAL utilities, and are introduced here.

-of *format* Select the format to create the new file as. The formats are assigned short names such as GTiff (for GeoTIFF) or HFA (for Erdas Imagine). The list of all format codes can be listed with the **--formats** switch. Only formats list as "(rw)" (read-write) can be written.

Many utilities default to creating GeoTIFF files if a format is not specified. File extensions are not used to guess output format, nor are extensions generally added by GDAL if not indicated in the filename by the user.

-co *NAME=VALUE* Many formats have one or more optional creation options that can be used to control particulars about the file created. For instance, the GeoTIFF driver supports creation options to control compression, and whether the file should be tiled.

The creation options available vary by format driver, and some simple formats have no creation options at all. A list of options supported for a format can be listed with the "--format <format>" command line option but the web page for the format is the definitive source of information on driver creation options. See `format specific documentation for legal creation options for each format`

-a_srs *SRS* Several utilities, (gdal_translate and gdalwarp) include the ability to specify coordinate systems with command line options like **-a_srs** (assign SRS to output), **-s_srs** (source SRS) and **-t_srs** (target SRS).

These utilities allow the coordinate system (SRS = spatial reference system) to be assigned in a variety of formats.

- **NAD27**/**NAD83**/**WGS84**/**WGS72**: These common geographic (lat/long) coordinate systems can be used directly by these names.
- **EPSG:***n*: Coordinate systems (projected or geographic) can be selected based on their EPSG codes, for instance EPSG:27700 is the British National Grid. A list of EPSG coordinate systems can be found in the GDAL data files gcs.csv and pcs.csv.
- *PROJ.4 Definitions*: A PROJ.4 definition string can be used as a coordinate system. For instance "+proj=utm +zone=11 +datum=WGS84". Take care to keep the proj.4 string together as a single argument to the command (usually by double quoting).
- *OpenGIS Well Known Text*: The Open GIS Consortium has defined a textual format for describing coordinate systems as part of the Simple Features specifications. This format is the internal working format for coordinate systems used in GDAL. The name of a file containing a WKT coordinate system definition may be used a coordinate system argument, or the entire coordinate system itself may be used as a command line option (though escaping all the quotes in WKT is quite challenging).
- *ESRI Well Known Text*: ESRI uses a slight variation on OGC WKT format in their ArcGIS product (ArcGIS .prj files), and these may be used in a similar manner to WKT files, but the filename should be prefixed with **ESRI::**. For example **"ESRI::NAD 1927 StatePlane Wyoming West FIPS 4904.prj"**.
- *Spatial References from URLs*: For example `http://spatialreference.org/ref/user/north-pacific-`
- *filename*: The name of a file containing WKT, PROJ.4 strings, or XML/GML coordinate system definitions can be provided.

1.2 General Command Line Switches

All GDAL command line utility programs support the following "general" options.

--version Report the version of GDAL and exit.

--formats List all raster formats supported by this GDAL build (read-only and read-write) and exit. The format support is indicated as follows: 'ro' is read-only driver; 'rw' is read or write (i.e. supports CreateCopy); 'rw+' is read, write and update (i.e. supports Create). A 'v' is appended for formats supporting virtual IO (/vsimem, /vsigzip, /vsizip, etc). A 's' is appended for formats supporting subdatasets. Note: The valid formats for the output of gdalwarp are formats that support the Create() method (marked as rw+), not just the CreateCopy() method.

--format *format* List detailed information about a single format driver. The *format* should be the short name reported in the **--formats** list, such as GTiff.

--optfile *file* Read the named file and substitute the contents into the command line options list. Lines beginning with # will be ignored. Multi-word arguments may be kept together with double quotes.

--config *key value* Sets the named `configuration keyword` to the given value, as opposed to setting them as environment variables. Some common configuration keywords are GDAL_CACHEMAX (memory used internally for caching in megabytes) and GDAL_DATA (path of the GDAL "data" directory). Individual drivers may be influenced by other configuration options.

--debug *value* Control what debugging messages are emitted. A value of *ON* will enable all debug messages. A value of *OFF* will disable all debug messages. Another value will select only debug messages containing that string in the debug prefix code.

--help-general Gives a brief usage message for the generic GDAL command line options and exit.

Chapter 2

gdalinfo

Lists information about a raster dataset.

2.1 SYNOPSIS

```
gdalinfo [--help-general] [-json] [-mm] [-stats] [-hist] [-nogcp] [-nomd]
         [-norat] [-noct] [-nofl] [-checksum] [-proj4]
         [-listmdd] [-mdd domain|`all`]*
         [-sd subdataset] [-oo NAME=VALUE]* datasetname
```

2.2 DESCRIPTION

The gdalinfo program lists various information about a GDAL supported raster dataset.

-json Display the output in json format.

-mm Force computation of the actual min/max values for each band in the dataset.

-stats Read and display image statistics. Force computation if no statistics are stored in an image.

-approx_stats Read and display image statistics. Force computation if no statistics are stored in an image. However, they may be computed based on overviews or a subset of all tiles. Useful if you are in a hurry and don't want precise stats.

-hist Report histogram information for all bands.

-nogcp Suppress ground control points list printing. It may be useful for datasets with huge amount of GCPs, such as L1B AVHRR or HDF4 MODIS which contain thousands of them.

-nomd Suppress metadata printing. Some datasets may contain a lot of metadata strings.

-norat Suppress printing of raster attribute table.

-noct Suppress printing of color table.

-checksum Force computation of the checksum for each band in the dataset.

-listmdd (GDAL >= 1.11) List all metadata domains available for the dataset.

-mdd domain Report metadata for the specified domain. Starting with GDAL 1.11, "all" can be used to report metadata in all domains

-nofl (GDAL >= 1.9.0) Only display the first file of the file list.

-sd *subdataset* (GDAL >= 1.9.0) If the input dataset contains several subdatasets read and display a subdataset with specified number (starting from 1). This is an alternative of giving the full subdataset name.

-proj4 (GDAL >= 1.9.0) Report a PROJ.4 string corresponding to the file's coordinate system.

-oo *NAME=VALUE*: (starting with GDAL 2.0) Dataset open option (format specific)

The gdalinfo will report all of the following (if known):

- The format driver used to access the file.
- Raster size (in pixels and lines).
- The coordinate system for the file (in OGC WKT).
- The geotransform associated with the file (rotational coefficients are currently not reported).
- Corner coordinates in georeferenced, and if possible lat/long based on the full geotransform (but not GCPs).
- Ground control points.
- File wide (including subdatasets) metadata.
- Band data types.
- Band color interpretations.
- Band block size.
- Band descriptions.
- Band min/max values (internally known and possibly computed).
- Band checksum (if computation asked).
- Band NODATA value.
- Band overview resolutions available.
- Band unit type (i.e.. "meters" or "feet" for elevation bands).
- Band pseudo-color tables.

2.3 C API

Starting with GDAL 2.1, this utility is also callable from C with GDALInfo().

2.4 EXAMPLE

```
gdalinfo ~/openev/utm.tif
Driver: GTiff/GeoTIFF
Size is 512, 512
Coordinate System is:
PROJCS["NAD27 / UTM zone 11N",
    GEOGCS["NAD27",
        DATUM["North_American_Datum_1927",
            SPHEROID["Clarke 1866",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)
Pixel Size = (60.000000,-60.000000)
```

```
Corner Coordinates:
Upper Left  (  440720.000, 3751320.000) (117d38'28.21"W, 33d54'8.47"N)
Lower Left  (  440720.000, 3720600.000) (117d38'20.79"W, 33d37'31.04"N)
Upper Right (  471440.000, 3751320.000) (117d18'32.07"W, 33d54'13.08"N)
Lower Right (  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)
Band 1 Block=512x16 Type=Byte, ColorInterp=Gray
```

Chapter 3

gdal_translate

Converts raster data between different formats.

3.1 SYNOPSIS

```
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[%]|0 ysize[%]|0] [-tr xres yres]
       [-r {nearest,bilinear,cubic,cubicspline,lanczos,average,mode}]
       [-unscale] [-scale[_bn] [src_min src_max [dst_min dst_max]]]* [-exponent[_bn] exp_val]*
       [-srcwin xoff yoff xsize ysize] [-epo] [-eco]
       [-projwin ulx uly lrx lry] [-projwin_srs srs_def]
       [-a_srs srs_def] [-a_ullr ulx uly lrx lry] [-a_nodata value]
       [-gcp pixel line easting northing [elevation]]*
       [-mo "META-TAG=VALUE"]* [-q] [-sds]
       [-co "NAME=VALUE"]* [-stats] [-norat]
       [-oo NAME=VALUE]*
       src_dataset dst_dataset
```

3.2 DESCRIPTION

The gdal_translate utility can be used to convert raster data between different formats, potentially performing some operations like subsettings, resampling, and rescaling pixels in the process.

-ot: ***type*** For the output bands to be of the indicated data type.

-strict: Don't be forgiving of mismatches and lost data when translating to the output format.

-of ***format*****:** Select the output format. The default is GeoTIFF (GTiff). Use the short format name.

-b ***band*****:** Select an input band *band* for output. Bands are numbered from 1. Multiple **-b** switches may be used to select a set of input bands to write to the output file, or to reorder bands. Starting with GDAL 1.8.0, *band* can also be set to "mask,1" (or just "mask") to mean the mask band of the first band of the input dataset.

-mask ***band*****:** (GDAL >= 1.8.0) Select an input band *band* to create output dataset mask band. Bands are numbered from 1. *band* can be set to "none" to avoid copying the global mask of the input dataset if it exists. Otherwise it is copied by default ("auto"), unless the mask is an alpha channel, or if it is explicitly used to be a regular band of the output dataset ("-b mask"). *band* can also be set to "mask,1" (or just "mask") to mean the mask band of the 1st band of the input dataset.

-expand ***gray*****|*****rgb*****|*****rgba*****:** (From GDAL 1.6.0) To expose a dataset with 1 band with a color table as a dataset with 3 (RGB) or 4 (RGBA) bands. Useful for output drivers such as JPEG, JPEG2000, MrSID, ECW that don't

support color indexed datasets. The 'gray' value (from GDAL 1.7.0) enables to expand a dataset with a color table that only contains gray levels to a gray indexed dataset.

-outsize *xsize[%]|0 ysize[%]|0*: Set the size of the output file. Outsize is in pixels and lines unless '%' is attached in which case it is as a fraction of the input image size. Starting with GDAL 2.0, if one of the 2 values is set to 0, its value will be determined from the other one, while maintaining the aspect ratio of the source dataset.

-tr xres yres : (starting with GDAL 2.0) set target resolution. The values must be expressed in georeferenced units. Both must be positive values. This is mutually exclusive with -outsize and -a_ullr.

-r *{nearest (default),bilinear,cubic,cubicspline,lanczos,average,mode}*: (GDAL >= 2.0) Select a resampling algorithm.

-scale *[src_min src_max [dst_min dst_max]]*: Rescale the input pixels values from the range *src_min* to *src_max* to the range *dst_min* to *dst_max*. If omitted the output range is 0 to 255. If omitted the input range is automatically computed from the source data. Before GDAL 1.11, it can be specified only once, and in that case, it applies to all bands of the output dataset. Starting with GDAL 1.11, -scale can be repeated several times (if specified only once, it also applies to all bands of the output dataset), so as to specify per band parameters. It is also possible to use the "-scale_bn" syntax where bn is a band number (e.g. "-scale_2" for the 2nd band of the output dataset) to specify the parameters of one or several specific bands.

-exponent *exp_val*: (From GDAL 1.11) To apply non-linear scaling with a power function. exp_val is the exponent of the power function (must be positive). This option must be used with the -scale option. If specified only once, -exponent applies to all bands of the output image. It can be repeated several times so as to specify per band parameters. It is also possible to use the "-exponent_bn" syntax where bn is a band number (e.g. "-exponent_2" for the 2nd band of the output dataset) to specify the parameters of one or several specific bands.

-unscale: Apply the scale/offset metadata for the bands to convert scaled values to unscaled values. It is also often necessary to reset the output datatype with the **-ot** switch.

-srcwin *xoff yoff xsize ysize*: Selects a subwindow from the source image for copying based on pixel/line location.

-projwin *ulx uly lrx lry*: Selects a subwindow from the source image for copying (like **-srcwin**) but with the corners given in georeferenced coordinates (by default expressed in the SRS of the dataset. Can be changed with -projwin_srs). Note: in GDAL 2.1.0 and 2.1.1, using -projwin with coordinates not aligned with pixels will result in a sub-pixel shift. This has been corrected in later versions. When selecting non-nearest neighbour resampling, starting with GDAL 2.1.0, sub-pixel accuracy is however used to get better results.

-projwin_srs *srs_def*: (GDAL >= 2.0) Specifies the SRS in which to interpret the coordinates given with -projwin. The *srs_def* may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT. Note that this does not cause reprojection of the dataset to the specified SRS.

-epo: (Error when Partially Outside) (GDAL >= 1.10) If this option is set, **-srcwin** or **-projwin** values that falls partially outside the source raster extent will be considered as an error. The default behaviour starting with GDAL 1.10 is to accept such requests, when they were considered as an error before.

-eco: (Error when Completely Outside) (GDAL >= 1.10) Same as **-epo**, except that the criterion for erroring out is when the request falls completely outside the source raster extent.

-a_srs *srs_def*: Override the projection for the output file. The *srs_def* may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.

-a_ullr *ulx uly lrx lry*: Assign/override the georeferenced bounds of the output file. This assigns georeferenced bounds to the output file, ignoring what would have been derived from the source file. So this does not cause reprojection to the specified SRS.

-a_nodata *value*: Assign a specified nodata value to output bands. Starting with GDAL 1.8.0, can be set to *none* to avoid setting a nodata value to the output file if one exists for the source file. Note that, if the input dataset has a nodata value, this does not cause pixel values that are equal to that nodata value to be changed to the value specified with this option.

-mo *"META-TAG=VALUE"*: Passes a metadata key and value to set on the output dataset if possible.

-co *"NAME=VALUE"*: Passes a creation option to the output format driver. Multiple **-co** options may be listed. See format specific documentation for legal creation options for each format.

-gcp *pixel line easting northing elevation*: Add the indicated ground control point to the output dataset. This option may be provided multiple times to provide a set of GCPs.

-q: Suppress progress monitor and other non-error output.

-sds: Copy all subdatasets of this file to individual output files. Use with formats like HDF or OGDI that have subdatasets. The output file naming scheme has changed in GDAL 1.11 (e.g. ofile_1.tif, ofile_2.tif).

-stats: (GDAL >= 1.8.0) Force (re)computation of statistics.

-norat (GDAL >= 1.11) Do not copy source RAT into destination dataset.

-oo *NAME=VALUE*: (starting with GDAL 2.0) Dataset open option (format specific)

***src_dataset*:** The source dataset name. It can be either file name, URL of data source or subdataset name for multi-dataset files.

***dst_dataset*:** The destination file name.

3.3 C API

Starting with GDAL 2.1, this utility is also callable from C with GDALTranslate().

3.4 EXAMPLE

```
gdal_translate -of GTiff -co "TILED=YES" utm.tif utm_tiled.tif
```

Starting with GDAL 1.8.0, to create a JPEG-compressed TIFF with internal mask from a RGBA dataset :

```
gdal_translate rgba.tif withmask.tif -b 1 -b 2 -b 3 -mask 4 -co COMPRESS=JPEG -co PHOTOMETRIC=YCBCR --config G
```

Starting with GDAL 1.8.0, to create a RGBA dataset from a RGB dataset with a mask :

```
gdal_translate withmask.tif rgba.tif -b 1 -b 2 -b 3 -b mask
```

Chapter 4

gdaladdo

Builds or rebuilds overview images.

4.1 SYNOPSIS

```
gdaladdo [-r {nearest,average,gauss,cubic,cubicspline,lanczos,average_mp,average_magphase,mode}]
         [-b band]*
         [-ro] [-clean] [-oo NAME=VALUE]* [--help-general] filename levels
```

4.2 DESCRIPTION

The gdaladdo utility can be used to build or rebuild overview images for most supported file formats with one of several downsampling algorithms.

-r ***{nearest (default),average,gauss,cubic,cubicspline,lanczos,average_mp,average_magphase,mode}*****:** Select a resampling algorithm.

-b ***band*****:** (available from GDAL 1.10) Select an input band *band* for overview generation. Band numbering starts from 1. Multiple **-b** switches may be used to select a set of input bands to generate overviews.

-ro: (available from GDAL 1.6.0) open the dataset in read-only mode, in order to generate external overview (for GeoTIFF especially).

-clean: (available from GDAL 1.7.0) remove all overviews.

-oo ***NAME=VALUE*****:** (starting with GDAL 2.0) Dataset open option (format specific)

filename**:** The file to build overviews for (or whose overviews must be removed).

levels**:** A list of integral overview levels to build. Ignored with -clean option.

Mode (available from GDAL 1.6.0) selects the value which appears most often of all the sampled points. *average_mp* is unsuitable for use. *Average_magphase* averages complex data in mag/phase space. *Nearest* and *average* are applicable to normal image data. *Nearest* applies a nearest neighbour (simple sampling) resampler, while *average* computes the average of all non-NODATA contributing pixels. *Cubic* resampling (available from GDAL 1.7.0) applies a cubic convolution kernel. *Gauss* resampling (available from GDAL 1.6.0) applies a Gaussian kernel before computing the overview, which can lead to better results than simple averaging in e.g case of sharp edges with high contrast or noisy patterns. The advised level values should be 2, 4, 8, ... so that a 3x3 resampling Gaussian kernel is selected. *CubicSpline* resampling (available from GDAL 2.0) applies a B-Spline convolution kernel. *Lanczos* resampling (available from GDAL 2.0) applies a Lanczos windowed sinc convolution kernel.

gdaladdo will honour properly NODATA_VALUES tuples (special dataset metadata) so that only a given RGB triplet (in case of a RGB image) will be considered as the nodata value and not each value of the triplet independently per band.

Selecting a level value like *2* causes an overview level that is 1/2 the resolution (in each dimension) of the base layer to be computed. If the file has existing overview levels at a level selected, those levels will be recomputed and rewritten in place.

For internal GeoTIFF overviews (or external overviews in GeoTIFF format), note that -clean does not shrink the file. A later run of gdaladdo with overview levels will cause the file to be expanded, rather than reusing the space of the previously deleted overviews. If you just want to change the resampling method on a file that already has overviews computed, you don't need to clean the existing overviews.

Some format drivers do not support overviews at all. Many format drivers store overviews in a secondary file with the extension .ovr that is actually in TIFF format. By default, the GeoTIFF driver stores overviews internally to the file operated on (if it is writable), unless the -ro flag is specified.

Most drivers also support an alternate overview format using Erdas Imagine format. To trigger this use the USE_RRD=YES configuration option. This will place the overviews in an associated .aux file suitable for direct use with Imagine or ArcGIS as well as GDAL applications. (e.g. –config USE_RRD YES)

4.3 External overviews in GeoTIFF format

External overviews created in TIFF format may be compressed using the COMPRESS_OVERVIEW configuration option. All compression methods, supported by the GeoTIFF driver, are available here. (e.g. –config COMPRESS_OVERVIEW DEFLATE). The photometric interpretation can be set with –config PHOTOMETRIC_OVERVIEW {RGB,YCBCR,...}, and the interleaving with –config INTERLEAVE_OVERVIEW {PIXEL|BAND}.

For JPEG compressed external overviews, the JPEG quality can be set with "--config JPEG_QUALITY_OVERVIEW value" (GDAL 1.7.0 or later).

For LZW or DEFLATE compressed external overviews, the predictor value can be set with "--config PREDICTOR_OVERVIEW 1|2|3" (GDAL 1.8.0 or later).

To produce the smallest possible JPEG-In-TIFF overviews, you should use :

```
--config COMPRESS_OVERVIEW JPEG --config PHOTOMETRIC_OVERVIEW YCBCR --config INTERLEAVE_OVERVIEW PIXEL
```

Starting with GDAL 1.7.0, external overviews can be created in the BigTIFF format by using the BIGTIFF_OVERVIEW configuration option : –config BIGTIFF_OVERVIEW {IF_NEEDED|IF_SAFER|YES|NO}. The default value is IF_NEEDED. The behaviour of this option is exactly the same as the BIGTIFF creation option documented in the GeoTIFF driver documentation.

- YES forces BigTIFF.
- NO forces classic TIFF.
- IF_NEEDED will only create a BigTIFF if it is clearly needed (uncompressed, and overviews larger than 4GB).
- IF_SAFER will create BigTIFF if the resulting file *might* exceed 4GB.

See the documentation of the GeoTIFF driver for further explanations on all those options.

4.4 C API

Functionality of this utility can be done from C with GDALBuildOverviews().

4.5 EXAMPLE

Create overviews, embedded in the supplied TIFF file:

```
gdaladdo -r average abc.tif 2 4 8 16
```

Create an external compressed GeoTIFF overview file from the ERDAS .IMG file:

```
gdaladdo -ro --config COMPRESS_OVERVIEW DEFLATE erdas.img 2 4 8 16
```

Create an external JPEG-compressed GeoTIFF overview file from a 3-band RGB dataset (if the dataset is a writable GeoTIFF, you also need to add the -ro option to force the generation of external overview):

```
gdaladdo --config COMPRESS_OVERVIEW JPEG --config PHOTOMETRIC_OVERVIEW YCBCR
         --config INTERLEAVE_OVERVIEW PIXEL rgb_dataset.ext 2 4 8 16
```

Create an Erdas Imagine format overviews for the indicated JPEG file:

```
gdaladdo --config USE_RRD YES airphoto.jpg 3 9 27 81
```

Chapter 5

gdaltindex

Builds a shapefile as a raster tileindex.

5.1 SYNOPSIS

```
gdaltindex [-f format] [-tileindex field_name] [-write_absolute_path]
           [-skip_different_projection] [-t_srs target_srs]
           [-src_srs_name field_name] [-src_srs_format [AUTO|WKT|EPSG|PROJ]
           [-lyr_name name] index_file [gdal_file]*
```

5.2 DESCRIPTION

This program builds a shapefile with a record for each input raster file, an attribute containing the filename, and a polygon geometry outlining the raster. This output is suitable for use with `MapServer` as a raster tileindex.

-f format: (GDAL >= 1.11)

The OGR format of the output tile index file. Default is ESRI Shapefile.

-tileindex field_name: The output field name to hold the file path/location to the indexed rasters. The default tile index field name is `location`.

-write_absolute_path: The absolute path to the raster files is stored in the tile index file. By default the raster filenames will be put in the file exactly as they are specified on the command line.

-skip_different_projection: Only files with same projection as files already inserted in the tileindex will be inserted (unless `-t_srs` is specified). Default does not check projection and accepts all inputs.

-t_srs target_srs: Geometries of input files will be transformed to the desired target coordinate reference system. Using this option generates files that are not compatible with MapServer < 6.4. Default creates simple rectangular polygons in the same coordinate reference system as the input rasters.

-src_srs_name field_name: (GDAL >= 1.11)

The name of the field to store the SRS of each tile. This field name can be used as the value of the TILESRS keyword in MapServer >= 6.4.

-src_srs_format type: (GDAL >= 1.11)

The format in which the SRS of each tile must be written. Types can be AUTO, WKT, EPSG, PROJ.

-lyr_name name: Layer name to create/append to in the output tile index file.

index_file: The name of the output file to create/append to. The default shapefile will be created if it doesn't already exist, otherwise it will append to the existing file.

gdal_file: The input GDAL raster files, can be multiple files separated by spaces. Wildcards my also be used. Stores the file locations in the same style as specified here, unless `-write_absolute_path` option is also used.

5.3 EXAMPLES

Produce a shapefile (`doq_index.shp`) with a record for every image that the utility found in the `doq` folder. Each record holds information that points to the location of the image and also a bounding rectangle shape showing the bounds of the image:

```
gdaltindex doq_index.shp doq/*.tif
```

The `-t_srs` option can also be used to transform all input rasters into the same output projection:

```
gdaltindex -t_srs EPSG:4326 -src_srs_name src_srs tile_index_mixed_srs.shp *.tif
```

Chapter 6

gdalbuildvrt

Builds a VRT from a list of datasets.

6.1 SYNOPSIS

```
gdalbuildvrt [-tileindex field_name]
             [-resolution {highest|lowest|average|user}]
             [-te xmin ymin xmax ymax] [-tr xres yres] [-tap]
             [-separate] [-b band]* [-sd subdataset]
             [-allow_projection_difference] [-q]
             [-addalpha] [-hidenodata]
             [-srcnodata "value [value...]"] [-vrtnodata "value [value...]"]
             [-a_srs srs_def]
             [-r {nearest,bilinear,cubic,cubicspline,lanczos,average,mode}]
             [-oo NAME=VALUE]*
             [-input_file_list my_list.txt] [-overwrite] output.vrt [gdalfile]*
```

6.2 DESCRIPTION

This program builds a VRT (Virtual Dataset) that is a mosaic of the list of input GDAL datasets. The list of input GDAL datasets can be specified at the end of the command line, or put in a text file (one filename per line) for very long lists, or it can be a MapServer tileindex (see gdaltindex utility). In the later case, all entries in the tile index will be added to the VRT.

With -separate, each files goes into a separate band in the VRT band. Otherwise, the files are considered as tiles of a larger mosaic and the VRT file has as many bands as one of the input files.

If one GDAL dataset is made of several subdatasets and has 0 raster bands, all the subdatasets will be added to the VRT rather than the dataset itself.

gdalbuildvrt does some amount of checks to assure that all files that will be put in the resulting VRT have similar characteristics : number of bands, projection, color interpretation... If not, files that do not match the common characteristics will be skipped. (This is only true in the default mode, and not when using the -separate option)

If there is some amount of spatial overlapping between files, the order of files appearing in the list of source matter: files that are listed at the end are the ones from which the content will be fetched. Note that nodata will be taken into account to potentially fetch data from less prioritary datasets, but currently, alpha channel is not taken into account to do alpha compositing (so a source with alpha=0 appearing on top of another source will override is content). This might be changed in later versions.

This utility is somehow equivalent to the gdal_vrtmerge.py utility and is build by default in GDAL 1.6.1.

-tileindex: Use the specified value as the tile index field, instead of the default value with is 'location'.

-resolution {highest|lowest|average|user}: In case the resolution of all input files is not the same, the -resolution

flag enables the user to control the way the output resolution is computed. 'average' is the default. 'highest' will pick the smallest values of pixel dimensions within the set of source rasters. 'lowest' will pick the largest values of pixel dimensions within the set of source rasters. 'average' will compute an average of pixel dimensions within the set of source rasters. 'user' is new in GDAL 1.7.0 and must be used in combination with the -tr option to specify the target resolution.

-tr xres yres : (starting with GDAL 1.7.0) set target resolution. The values must be expressed in georeferenced units. Both must be positive values. Specifying those values is of course incompatible with highest|lowest|average values for -resolution option.

-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.

-te xmin ymin xmax ymax : (starting with GDAL 1.7.0) set georeferenced extents of VRT file. The values must be expressed in georeferenced units. If not specified, the extent of the VRT is the minimum bounding box of the set of source rasters.

-addalpha: (starting with GDAL 1.7.0) Adds an alpha mask band to the VRT when the source raster have none. Mainly useful for RGB sources (or grey-level sources). The alpha band is filled on-the-fly with the value 0 in areas without any source raster, and with value 255 in areas with source raster. The effect is that a RGBA viewer will render the areas without source rasters as transparent and areas with source rasters as opaque. This option is not compatible with -separate.

-hidenodata: (starting with GDAL 1.7.0) Even if any band contains nodata value, giving this option makes the VRT band not report the NoData. Useful when you want to control the background color of the dataset. By using along with the -addalpha option, you can prepare a dataset which doesn't report nodata value but is transparent in areas with no data.

-srcnodata *value [value...]*: (starting with GDAL 1.7.0) Set nodata values for input bands (different values can be supplied for each band). If more than one value is supplied all values should be quoted to keep them together as a single operating system argument. If the option is not specified, the intrinsic nodata settings on the source datasets will be used (if they exist). The value set by this option is written in the NODATA element of each ComplexSource element. Use a value of `None` to ignore intrinsic nodata settings on the source datasets.

-b *band*: (GDAL >= 1.10.0) Select an input *band* to be processed. Bands are numbered from 1. If input bands not set all bands will be added to vrt. Multiple **-b** switches may be used to select a set of input bands.

-sd *subdataset* (GDAL >= 1.10.0) If the input dataset contains several subdatasets use a subdataset with the specified number (starting from 1). This is an alternative of giving the full subdataset name as an input.

-vrtnodata *value [value...]*: (starting with GDAL 1.7.0) Set nodata values at the VRT band level (different values can be supplied for each band). If more than one value is supplied all values should be quoted to keep them together as a single operating system argument. If the option is not specified, intrinsic nodata settings on the first dataset will be used (if they exist). The value set by this option is written in the NoDataValue element of each VRTRasterBand element. Use a value of `None` to ignore intrinsic nodata settings on the source datasets.

-separate: (starting with GDAL 1.7.0) Place each input file into a separate band. In that case, only the first band of each dataset will be placed into a new band. Contrary to the default mode, it is not required that all bands have the same datatype.

-allow_projection_difference: (starting with GDAL 1.7.0) When this option is specified, the utility will accept to make a VRT even if the input datasets have not the same projection. Note: this does not mean that they will be reprojected. Their projection will just be ignored.

-a_srs *srs_def*: (starting with GDAL 1.10) Override the projection for the output file. The *srs_def* may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.

-r *{nearest (default),bilinear,cubic,cubicspline,lanczos,average,mode}*: (GDAL >= 2.0) Select a resampling algorithm.

-oo *NAME=VALUE*: (starting with GDAL 2.2) Dataset open option (format specific)

-input_file_list: To specify a text file with an input filename on each line

-q: (starting with GDAL 1.7.0) To disable the progress bar on the console

-overwrite: Overwrite the VRT if it already exists.

6.3 EXAMPLE

Make a virtual mosaic from all TIFF files contained in a directory :

```
gdalbuildvrt doq_index.vrt doq/*.tif
```

Make a virtual mosaic from files whose name is specified in a text file :

```
gdalbuildvrt -input_file_list my_list.txt doq_index.vrt
```

Make a RGB virtual mosaic from 3 single-band input files :

```
gdalbuildvrt -separate rgb.vrt red.tif green.tif blue.tif
```

Make a virtual mosaic with blue background colour (RGB: 0 0 255) :

```
gdalbuildvrt -hidenodata -vrtnodata "0 0 255" doq_index.vrt doq/*.tif
```

Chapter 7

gdal_contour

Builds vector contour lines from a raster elevation model.

7.1 SYNOPSIS

```
Usage: gdal_contour [-b <band>] [-a <attribute_name>] [-3d] [-inodata]
                    [-snodata n] [-i <interval>]
                    [-f <formatname>] [[-dsco NAME=VALUE] ...] [[-lco NAME=VALUE] ...]
                    [-off <offset>] [-fl <level> <level>...]
                    [-nln <outlayername>]
                    <src_filename> <dst_filename>
```

7.2 DESCRIPTION

This program generates a vector contour file from the input raster elevation model (DEM).

Starting from version 1.7 the contour line-strings will be oriented consistently. The high side will be on the right, i.e. a line string goes clockwise around a top.

-b *band*: picks a particular band to get the DEM from. Defaults to band 1.

-a *name*: provides a name for the attribute in which to put the elevation. If not provided no elevation attribute is attached.

-3d: Force production of 3D vectors instead of 2D. Includes elevation at every vertex.

-inodata: Ignore any nodata value implied in the dataset - treat all values as valid.

-snodata *value*: Input pixel value to treat as "nodata".

-f *format*: create output in a particular format, default is shapefiles.

-dsco *NAME=VALUE*: Dataset creation option (format specific)

-lco *NAME=VALUE*: Layer creation option (format specific)

-i *interval*: elevation interval between contours.

-off *offset*: Offset from zero relative to which to interpret intervals.

-fl *level*: Name one or more "fixed levels" to extract.

-nln *outlayername*: Provide a name for the output vector layer. Defaults to "contour".

7.3 C API

Functionality of this utility can be done from C with GDALContourGenerate().

7.4 EXAMPLE

This would create 10meter contours from the DEM data in dem.tif and produce a shapefile in contour.shp/shx/dbf with the contour elevations in the "elev" attribute.

```
gdal_contour -a elev dem.tif contour.shp -i 10.0
```

Chapter 8

gdal_rasterize

Burns vector geometries into a raster.

8.1 SYNOPSIS

```
Usage: gdal_rasterize [-b band]* [-i] [-at]
       {[-burn value]* | [-a attribute_name] | [-3d]} [-add]
       [-l layername]* [-where expression] [-sql select_statement]
       [-dialect dialect] [-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>
```

8.2 DESCRIPTION

This program burns vector geometries (points, lines, and polygons) into the raster band(s) of a raster image. Vectors are read from OGR supported vector formats.

Note that on the fly reprojection of vector data to the coordinate system of the raster data is only supported since GDAL 2.1.0.

Since GDAL 1.8.0, the target GDAL file can be created by gdal_rasterize. Either the -tr or -ts option must be used in that case.

-b *band*: The band(s) to burn values into. Multiple -b arguments may be used to burn into a list of bands. The default is to burn into band 1.

-i: Invert rasterization. Burn the fixed burn value, or the burn value associated with the first feature into all parts of the image *not* inside the provided polygon.

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

-burn *value*: A fixed value to burn into a band for all objects. A list of -burn options can be supplied, one per band being written to.

-a *attribute_name*: Identifies an attribute field on the features to be used for a burn-in value. The value will be burned into all output bands.

-3d: Indicates that a burn value should be extracted from the "Z" values of the feature. As of now, only points and lines are drawn in 3D.

-add: Instead of burning a new value, this adds the new value to the existing raster. Suitable for heatmaps for instance.

-l *layername*: Indicates the layer(s) from the datasource that will be used for input features. May be specified multiple times, but at least one layer name or a -sql option must be specified.

-where *expression*: An optional SQL WHERE style query expression to be applied to select features to burn in from the input layer(s).

-sql *select_statement*: An SQL statement to be evaluated against the datasource to produce a virtual layer of features to be burned in.

-dialect *dialect*: (GDAL >= 2.1.0) SQL dialect. In some cases can be used to use (unoptimized) OGR SQL instead of the native SQL of an RDBMS by passing OGRSQL. Starting with GDAL 1.10, the "SQLITE" dialect can also be used with any datasource.

-of *format*: (GDAL >= 1.8.0) Select the output format. The default is GeoTIFF (GTiff). Use the short format name.

-a_nodata *value*: (GDAL >= 1.8.0) Assign a specified nodata value to output bands.

-init *value*: (GDAL >= 1.8.0) Pre-initialize the output image bands with these values. However, it is not marked as the nodata value in the output file. If only one value is given, the same value is used in all the bands.

-a_srs *srs_def*: (GDAL >= 1.8.0) Override the projection for the output file. If not specified, the projection of the input vector file will be used if available. If incompatible projections between input and output files, no attempt will be made to reproject features. The *srs_def* may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.

-co *"NAME=VALUE"*: (GDAL >= 1.8.0) Passes a creation option to the output format driver. Multiple **-co** options may be listed. See `format specific documentation for legal creation options for each format`.

-te *xmin ymin xmax ymax* : (GDAL >= 1.8.0) Set georeferenced extents. The values must be expressed in georeferenced units. If not specified, the extent of the output file will be the extent of the vector layers.

-tr *xres yres* : (GDAL >= 1.8.0) Set target resolution. The values must be expressed in georeferenced units. Both must be positive values.

-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*: (GDAL >= 1.8.0) Set output file size in pixels and lines. Note that -ts cannot be used with -tr

-ot *type*: (GDAL >= 1.8.0) For the output bands to be of the indicated data type. Defaults to Float64

-q: (GDAL >= 1.8.0) Suppress progress monitor and other non-error output.

***src_datasource*:** Any OGR supported readable datasource.

***dst_filename*:** The GDAL supported output file. Must support update mode access. Before GDAL 1.8.0, gdal_rasterize could not create new output files.

8.3 C API

Starting with GDAL 2.1, this utility is also callable from C with GDALRasterize().

8.4 EXAMPLE

The following would burn all polygons from mask.shp into the RGB TIFF file work.tif with the color red (RGB = 255,0,0).

```
gdal_rasterize -b 1 -b 2 -b 3 -burn 255 -burn 0 -burn 0 -l mask mask.shp work.tif
```

The following would burn all "class A" buildings into the output elevation file, pulling the top elevation from the ROOF_H attribute.

```
gdal_rasterize -a ROOF_H -where 'class="A"' -l footprints footprints.shp city_dem.tif
```

Chapter 9

rgb2pct.py

Convert a 24bit RGB image to 8bit paletted.

9.1 SYNOPSIS

```
rgb2pct.py [-n colors | -pct palette_file] [-of format] source_file dest_file
```

9.2 DESCRIPTION

This utility will compute an optimal pseudo-color table for a given RGB image using a median cut algorithm on a downsampled RGB histogram. Then it converts the image into a pseudo-colored image using the color table. This conversion utilizes Floyd-Steinberg dithering (error diffusion) to maximize output image visual quality.

-n *colors*: Select the number of colors in the generated color table. Defaults to 256. Must be between 2 and 256.

-pct *palette_file*: Extract the color table from *palette_file* instead of computing it. Can be used to have a consistent color table for multiple files. The *palette_file* must be a raster file in a GDAL supported format with a palette.

-of *format*: Format to generated (defaults to GeoTIFF). Same semantics as the **-of** flag for gdal_translate. Only output formats supporting pseudo-color tables should be used.

***source_file*:** The input RGB file.

***dest_file*:** The output pseudo-colored file that will be created.

NOTE: rgb2pct.py is a Python script, and will only work if GDAL was built with Python support.

9.3 EXAMPLE

If it is desired to hand create the palette, likely the simplest text format is the GDAL VRT format. In the following example a VRT was created in a text editor with a small 4 color palette with the RGBA colors 238/238/238/255, 237/237/237/255, 236/236/236/255 and 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>
```

Chapter 10

pct2rgb.py

Convert an 8bit paletted image to 24bit RGB.

10.1 SYNOPSIS

```
pct2rgb.py [-of format] [-b band] [-rgba] source_file dest_file
```

10.2 DESCRIPTION

This utility will convert a pseudo-color band on the input file into an output RGB file of the desired format.

-of *format*: Format to generated (defaults to GeoTIFF).

-b *band*: Band to convert to RGB, defaults to 1.

-rgba: Generate a RGBA file (instead of a RGB file by default).

***source_file*:** The input file.

***dest_file*:** The output RGB file that will be created.

NOTE: pct2rgb.py is a Python script, and will only work if GDAL was built with Python support.
The new '-expand rgb|rgba' option of gdal_translate obsoletes that utility.

Chapter 11

gdaltransform

Transforms coordinates.

11.1 SYNOPSIS

```
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]]* [-output_xy]
    [srcfile [dstfile]]
```

11.2 DESCRIPTION

The gdaltransform utility reprojects a list of coordinates into any supported projection,including GCP-based transformations.

-s_srs *srs def*: source spatial reference set. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG PCS and GCSes (i.e. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prj file containing well known text.

-t_srs *srs_def*: target spatial reference set. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG PCS and GCSes (i.e. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prj file containing well known text.

-to *NAME=VALUE*: set a transformer option suitable to pass to GDALCreateGenImgProjTransformer2().

-order *n*: order of polynomial used for warping (1 to 3). The default is to select a polynomial order based on the number of GCPs.

-tps: Force use of thin plate spline transformer based on available GCPs.

-rpc: Force use of RPCs.

-geoloc: Force use of Geolocation Arrays.

-i Inverse transformation: from destination to source.

-gcp*pixel line easting northing [elevation]*: Provide a GCP to be used for transformation (generally three or more are required)

-output_xy: (GDAL >= 2.0) Restrict output to "x y" instead of "x y z"

***srcfile*:** File with source projection definition or GCP's. If not given, source projection is read from the command-line -s_srs or -gcp parameters

***dstfile*:** File with destination projection definition.

Coordinates are read as pairs (or triples) of numbers per line from standard input, transformed, and written out to standard output in the same way. All transformations offered by gdalwarp are handled, including gcp-based ones.

Note that input and output must always be in decimal form. There is currently no support for DMS input or output.

If an input image file is provided, input is in pixel/line coordinates on that image. If an output file is provided, output is in pixel/line coordinates on that image.

11.3 Reprojection Example

Simple reprojection from one projected coordinate system to another:

```
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
```

11.4 Image RPC Example

The following command requests an RPC based transformation using the RPC model associated with the named file. Because the -i (inverse) flag is used, the transformation is from output georeferenced (WGS84) coordinates back to image coordinates.

```
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
```

Chapter 12

nearblack

Convert nearly black/white borders to black.

12.1 SYNOPSIS

```
nearblack [-of format] [-white | [-color c1,c2,c3...cn]*] [-near dist] [-nb non_black_pixels]
          [-setalpha] [-setmask] [-o outfile] [-q]  [-co "NAME=VALUE"]* infile
```

12.2 DESCRIPTION

This utility will scan an image and try to set all pixels that are nearly or exactly black, white or one or more custom colors around the collar to black or white. This is often used to "fix up" lossy compressed air photos so that color pixels can be treated as transparent when mosaicing.

-o *outfile*: The name of the output file to be created. Newly created files are created with the HFA driver by default (Erdas Imagine - .img)

-of *format*: (GDAL 1.8.0 or later) Select the output format. Use the short format name (GTiff for GeoTIFF for example).

-co *"NAME=VALUE"*: (GDAL 1.8.0 or later) Passes a creation option to the output format driver. Multiple **-co** options may be listed. See format specific documentation for legal creation options for each format. Only valid when creating a new file

-white: Search for nearly white (255) pixels instead of nearly black pixels.

-color *c1,c2,c3...cn*: (GDAL >= 1.9.0) Search for pixels near the specified color. May be specified multiple times. When -color is specified, the pixels that are considered as the collar are set to 0.

-near *dist*: Select how far from black, white or custom colors the pixel values can be and still considered near black, white or custom color. Defaults to 15.

-nb *non_black_pixels*: number of non-black pixels that can be encountered before the giving up search inwards. Defaults to 2.

-setalpha: (GDAL 1.8.0 or later) Adds an alpha band if the output file is specified and the input file has 3 bands, or sets the alpha band of the output file if it is specified and the input file has 4 bands, or sets the alpha band of the input file if it has 4 bands and no output file is specified. The alpha band is set to 0 in the image collar and to 255 elsewhere.

-setmask: (GDAL 1.8.0 or later) Adds a mask band to the output file, or adds a mask band to the input file if it does not already have one and no output file is specified. The mask band is set to 0 in the image collar and to 255 elsewhere.

-q: (GDAL 1.8.0 or later) Suppress progress monitor and other non-error output.

***infile*:** The input file. Any GDAL supported format, any number of bands, normally 8bit Byte bands.

The algorithm processes the image one scanline at a time. A scan "in" is done from either end setting pixels to black or white until at least "non_black_pixels" pixels that are more than "dist" gray levels away from black, white or custom colors have been encountered at which point the scan stops. The nearly black, white or custom color pixels are set to black or white. The algorithm also scans from top to bottom and from bottom to top to identify indentations in the top or bottom.

The processing is all done in 8bit (Bytes).

If the output file is omitted, the processed results will be written back to the input file - which must support update.

12.3 C API

Starting with GDAL 2.1, this utility is also callable from C with GDALNearblack().

Chapter 13

gdal_merge.py

Mosaics a set of images.

13.1 SYNOPSIS

```
gdal_merge.py [-o out_filename] [-of out_format] [-co NAME=VALUE]*
              [-ps pixelsize_x pixelsize_y] [-tap] [-separate] [-q] [-v] [-pct]
              [-ul_lr ulx uly lrx lry] [-init "value [value...]"]
              [-n nodata_value] [-a_nodata output_nodata_value]
              [-ot datatype] [-createonly] input_files
```

13.2 DESCRIPTION

This utility will automatically mosaic a set of images. All the images must be in the same coordinate system and have a matching number of bands, but they may be overlapping, and at different resolutions. In areas of overlap, the last image will be copied over earlier ones.

-o *out_filename*: The name of the output file, which will be created if it does not already exist (defaults to "out.tif").

-of *format*: Output format, defaults to GeoTIFF (GTiff).

-co *NAME=VALUE*: Creation option for output file. Multiple options can be specified. See format specific documentation for legal creation options for each format

-ot *datatype*: Force the output image bands to have a specific type. Use type names (i.e. Byte, Int16,...)

-ps *pixelsize_x pixelsize_y*: Pixel size to be used for the output file. If not specified the resolution of the first input file will be used.

-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*: The extents of the output file. If not specified the aggregate extents of all input files will be used.

-v: Generate verbose output of mosaicing operations as they are done.

-separate: Place each input file into a separate band.

-pct: Grab a pseudo-color table from the first input image, and use it for the output. Merging pseudo-colored images this way assumes that all input files use the same color table.

-n *nodata_value*: Ignore pixels from files being merged in with this pixel value.

-a_nodata *output_nodata_value*: (GDAL >= 1.9.0) Assign a specified nodata value to output bands.

-init *"value(s)"*: Pre-initialize the output image bands with these values. However, it is not marked as the nodata value in the output file. If only one value is given, the same value is used in all the bands.

-createonly: The output file is created (and potentially pre-initialized) but no input image data is copied into it.

NOTE: gdal_merge.py is a Python script, and will only work if GDAL was built with Python support.

13.3 EXAMPLE

Create an image with the pixels in all bands initialized to 255.

```
% gdal_merge.py -init 255 -o out.tif in1.tif in2.tif
```

Create an RGB image that shows blue in pixels with no data. The first two bands will be initialized to 0 and the third band will be initialized to 255.

```
% gdal_merge.py -init "0 0 255" -o out.tif in1.tif in2.tif
```

Chapter 14

gdal2tiles.py

Generates directory with TMS tiles, KMLs and simple web viewers.

14.1 SYNOPSIS

```
gdal2tiles.py [-p profile] [-r resampling] [-s srs] [-z zoom]
              [-e] [-a nodata] [-v] [-q] [-h] [-k] [-n] [-u url]
              [-w webviewer] [-t title] [-c copyright]
              [-g googlekey] [-b bingkey] input_file [output_dir]
```

14.2 DESCRIPTION

This utility generates a directory with small tiles and metadata, following the OSGeo Tile Map Service Specification. Simple web pages with viewers based on Google Maps, OpenLayers and Leaflet are generated as well - so anybody can comfortably explore your maps on-line and you do not need to install or configure any special software (like MapServer) and the map displays very fast in the web browser. You only need to upload the generated directory onto a web server.

GDAL2Tiles also creates the necessary metadata for Google Earth (KML SuperOverlay), in case the supplied map uses EPSG:4326 projection.

World files and embedded georeferencing is used during tile generation, but you can publish a picture without proper georeferencing too.

-p *PROFILE*, –profile=*PROFILE*: Tile cutting profile (mercator,geodetic,raster) - default 'mercator' (Google Maps compatible).

-r *RESAMPLING*, –resampling=*RESAMPLING*: Resampling method (average,near,bilinear,cubic,cubicspline,lanczos,antialias) - default 'average'.

-s *SRS*, –s_srs=*SRS*: The spatial reference system used for the source input data.

-z *ZOOM*, –zoom=*ZOOM*: Zoom levels to render (format:'2-5' or '10').

-e, –resume: Resume mode. Generate only missing files.

-a *NODATA*, –srcnodata=*NODATA*: NODATA transparency value to assign to the input data.

-v, –verbose Generate verbose output of tile generation.

-q, –quiet Disable messages and status to stdout (GDAL >= 2.1).

-h, –help Show help message and exit.

--version Show program's version number and exit.

KML (Google Earth) options:

Options for generated Google Earth SuperOverlay metadata

-k, –force-kml Generate KML for Google Earth - default for 'geodetic' profile and 'raster' in EPSG:4326. For a dataset with different projection use with caution!

-n, –no-kml: Avoid automatic generation of KML files for EPSG:4326.

-u *URL*, –url=*URL*: URL address where the generated tiles are going to be published.

Web viewer options:

Options for generated HTML viewers a la Google Maps

-w *WEBVIEWER*, –webviewer=*WEBVIEWER*: Web viewer to generate (all,google,openlayers,leaflet,none) - default 'all'.

-t *TITLE*, –title=*TITLE*: Title of the map.

-c *COPYRIGHT*, –copyright=*COPYRIGHT*: Copyright for the map.

-g *GOOGLEKEY*, –googlekey=*GOOGLEKEY*: Google Maps API key from http://code.google.-com/apis/maps/signup.html.

-b *BINGKEY*, –bingkey=*BINGKEY*: Bing Maps API key from https://www.bingmapsportal.com/

NOTE: gdal2tiles.py is a Python script that needs to be run against "new generation" Python GDAL binding.

Chapter 15

gdal-config

Determines various information about a GDAL installation.

15.1 SYNOPSIS

```
gdal-config [OPTIONS]
Options:
        [--prefix[=DIR]]
        [--libs]
        [--cflags]
        [--version]
        [--ogr-enabled]
        [--formats]
```

15.2 DESCRIPTION

This utility script (available on Unix systems) can be used to determine various information about a GDAL installation. It is normally just used by configure scripts for applications using GDAL but can be queried by an end user.

--prefix: the top level directory for the GDAL installation.

--libs: The libraries and link directives required to use GDAL.

--cflags: The include and macro definition required to compiled modules using GDAL.

--version: Reports the GDAL version.

--ogr-enabled: Reports "yes" or "no" to standard output depending on whether OGR is built into GDAL.

--formats: Reports which formats are configured into GDAL to stdout.

Chapter 16

gdal_retile.py

Retiles a set of tiles and/or build tiled pyramid levels.

```
gdal_retile.py [-v] [-co NAME=VALUE]* [-of out_format] [-ps pixelWidth pixelHeight]
               [-overlap val_in_pixel]
               [-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
```

This utility will retile a set of input tile(s). All the input tile(s) must be georeferenced in the same coordinate system and have a matching number of bands. Optionally pyramid levels are generated. It is possible to generate shape file(s) for the tiled output.

If your number of input tiles exhausts the command line buffer, use the general –optfile option

-targetDir *directory*: The directory where the tile result is created. Pyramids are stored in sub-directories numbered from 1. Created tile names have a numbering schema and contain the name of the source tiles(s)

-of *format*: Output format, defaults to GeoTIFF (GTiff).

-co *NAME=VALUE*: Creation option for output file. Multiple options can be specified. See format specific documentation for legal creation options for each format

-ot *datatype*: Force the output image bands to have a specific type. Use type names (i.e. Byte, Int16,...)

-ps *pixelsize_x pixelsize_y*: Pixel size to be used for the output file. If not specified, 256 x 256 is the default

-overlap *val_in_pixel*: (GDAL >= 2.2) Overlap in pixels between consecutive tiles. If not specified, 0 is the default

-levels *numberOfLevels*: Number of pyramids levels to build.

-v: Generate verbose output of tile operations as they are done.

-pyramidOnly: No retiling, build only the pyramids

-r *algorithm*: Resampling algorithm, default is near

-s_srs *srs_def*: Source spatial reference to use. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG, PCS, and GCSes (i.e. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prj file containing well known text. If no srs_def is given, the srs_def of the source tiles is used (if there is any). The srs_def will be propagated to created tiles (if possible) and to the optional shape file(s)

-tileIndex *tileIndexName*: The name of shape file containing the result tile(s) index

-tileIndexField *tileIndexFieldName*: The name of the attribute containing the tile name

-csv *csvFileName*: The name of the csv file containing the tile(s) georeferencing information. The file contains 5 columns: tilename,minx,maxx,miny,maxy

-csvDelim *column delimiter*: The column delimiter used in the CSV file, default value is a semicolon ";"

-useDirForEachRow: Normally the tiles of the base image are stored as described in **-targetDir**. For large images, some file systems have performance problems if the number of files in a directory is to big, causing gdal_retile not to finish in reasonable time. Using this parameter creates a different output structure. The tiles of the base image are stored in a sub-directory called 0, the pyramids in sub-directories numbered 1,2,.... Within each of these directories another level of sub-directories is created, numbered from 0...n, depending of how many tile rows are needed for each level. Finally, a directory contains only the tiles for one row for a specific level. For large images a performance improvement of a factor N could be achieved.

NOTE: gdal_retile.py is a Python script, and will only work if GDAL was built with Python support.

Chapter 17

gdal_grid

Creates regular grid from the scattered data.

17.1 SYNOPSIS

```
gdal_grid [-ot {Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/
          CInt16/CInt32/CFloat32/CFloat64}]
          [-of format] [-co "NAME=VALUE"]
          [-zfield field_name] [-z_increase increase_value] [-z_multiply multiply_value]
          [-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>
```

17.2 DESCRIPTION

This program creates regular grid (raster) from the scattered data read from the OGR datasource. Input data will be interpolated to fill grid nodes with values, you can choose from various interpolation methods.

Starting with GDAL 1.10, it is possible to set the **GDAL_NUM_THREADS** configuration option to parallelize the processing. The value to specify is the number of worker threads, or *ALL_CPUS* to use all the cores/CPUs of the computer.

-ot *type*: For the output bands to be of the indicated data type.

-of *format*: Select the output format. The default is GeoTIFF (GTiff). Use the short format name.

-txe *xmin xmax*: Set georeferenced X extents of output file to be created.

-tye *ymin ymax*: Set georeferenced Y extents of output file to be created.

-outsize *xsize ysize*: Set the size of the output file in pixels and lines.

-a_srs *srs_def*: Override the projection for the output file. The *srs_def* may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.

-zfield *field_name*: Identifies an attribute field on the features to be used to get a Z value from. This value overrides Z value read from feature geometry record (naturally, if you have a Z value in geometry, otherwise you have no choice and should specify a field name containing Z value).

-z_increase *increase_value*: Addition to the attribute field on the features to be used to get a Z value from. The addition should be the same unit as Z value. The result value will be Z value + Z increase value. The default value is 0.

-z_multiply *multiply_value*: This is multiplication ratio for Z field. This can be used for shift from e.g. foot to meters or from elevation to deep. The result value will be (Z value + Z increase value) $*$ Z multiply value. The default value is 1.

-a *[algorithm[:parameter1=value1][:parameter2=value2]...]*: Set the interpolation algorithm or data metric name and (optionally) its parameters. See INTERPOLATION ALGORITHMS and DATA METRICS sections for further discussion of available options.

-spat *xmin ymin xmax ymax*: Adds a spatial filter to select only features contained within the bounding box described by (xmin, ymin) - (xmax, ymax).

-clipsrc*[xmin ymin xmax ymax]|WKT|datasource|spat_extent*: Adds a spatial filter to select only features contained within the specified bounding box (expressed in source SRS), WKT geometry (POLYGON or MULTIPOLYGON), from a datasource or to the spatial extent of the **-spat** option if you use the *spat_extent* keyword. When specifying a datasource, you will generally want to use it in combination of the **-clipsrclayer**, **-clipsrcwhere** or **-clipsrcsql** options.

-clipsrcsql *sql_statement*: Select desired geometries using an SQL query instead.

-clipsrclayer *layername*: Select the named layer from the source clip datasource.

-clipsrcwhere *expression*: Restrict desired geometries based on attribute query.

-l *layername*: Indicates the layer(s) from the datasource that will be used for input features. May be specified multiple times, but at least one layer name or a **-sql** option must be specified.

-where *expression*: An optional SQL WHERE style query expression to be applied to select features to process from the input layer(s).

-sql *select_statement*: An SQL statement to be evaluated against the datasource to produce a virtual layer of features to be processed.

-co *"NAME=VALUE"*: Passes a creation option to the output format driver. Multiple **-co** options may be listed. See `format specific documentation for legal creation options for each format`.

-q: Suppress progress monitor and other non-error output.

***src_datasource*:** Any OGR supported readable datasource.

***dst_filename*:** The GDAL supported output file.

17.3 INTERPOLATION ALGORITHMS

There are number of interpolation algorithms to choose from.

17.3.1 invdist

Inverse distance to a power. This is default algorithm. It has following parameters:

***power*:** Weighting power (default 2.0).

***smoothing*:** Smoothing parameter (default 0.0).

***radius1*:** The first radius (X axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.

radius2: The second radius (Y axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.

angle: Angle of search ellipse rotation in degrees (counter clockwise, default 0.0).

max_points: Maximum number of data points to use. Do not search for more points than this number. This is only used if search ellipse is set (both radii are non-zero). Zero means that all found points should be used. Default is 0.

min_points: Minimum number of data points to use. If less amount of points found the grid node considered empty and will be filled with NODATA marker. This is only used if search ellipse is set (both radii are non-zero). Default is 0.

nodata: NODATA marker to fill empty points (default 0.0).

17.3.2 invdistnn

(Since GDAL 2.1) Inverse distance to a power with nearest neighbor searching, ideal when max_points is used. It has following parameters:

power: Weighting power (default 2.0).

smoothing: Smoothing parameter (default 0.0).

radius: The radius of the search circle, which should be non-zero. Default is 1.0.

max_points: Maximum number of data points to use. Do not search for more points than this number. Found points will be ranked from nearest to furthest distance when weighting. Default is 12.

min_points: Minimum number of data points to use. If less amount of points found the grid node is considered empty and will be filled with NODATA marker. Default is 0.

nodata: NODATA marker to fill empty points (default 0.0).

17.3.3 average

Moving average algorithm. It has following parameters:

radius1: The first radius (X axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.

radius2: The second radius (Y axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.

angle: Angle of search ellipse rotation in degrees (counter clockwise, default 0.0).

min_points: Minimum number of data points to use. If less amount of points found the grid node considered empty and will be filled with NODATA marker. Default is 0.

nodata: NODATA marker to fill empty points (default 0.0).

Note, that it is essential to set search ellipse for moving average method. It is a window that will be averaged when computing grid nodes values.

17.3.4 nearest

Nearest neighbor algorithm. It has following parameters:

radius1: The first radius (X axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.

radius2: The second radius (Y axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.

angle: Angle of search ellipse rotation in degrees (counter clockwise, default 0.0).

nodata: NODATA marker to fill empty points (default 0.0).

17.3.5 linear

(Since GDAL 2.1) Linear interpolation algorithm.

The Linear method performs linear interpolation by computing a Delaunay triangulation of the point cloud, finding in which triangle of the triangulation the point is, and by doing linear interpolation from its barycentric coordinates within the triangle. If the point is not in any triangle, depending on the radius, the algorithm will use the value of the nearest point or the nodata value.

It has following parameters:

radius: In case the point to be interpolated does not fit into a triangle of the Delaunay triangulation, use that maximum distance to search a nearest neighbour, or use nodata otherwise. If set to -1, the search distance is infinite. If set to 0, nodata value will be always used. Default is -1.

nodata: NODATA marker to fill empty points (default 0.0).

17.4 DATA METRICS

Besides the interpolation functionality gdal_grid can be used to compute some data metrics using the specified window and output grid geometry. These metrics are:

minimum: Minimum value found in grid node search ellipse.

maximum: Maximum value found in grid node search ellipse.

range: A difference between the minimum and maximum values found in grid node search ellipse.

count: A number of data points found in grid node search ellipse.

average_distance: An average distance between the grid node (center of the search ellipse) and all of the data points found in grid node search ellipse.

average_distance_pts: An average distance between the data points found in grid node search ellipse. The distance between each pair of points within ellipse is calculated and average of all distances is set as a grid node value.

All the metrics have the same set of options:

radius1: The first radius (X axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.

radius2: The second radius (Y axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.

angle: Angle of search ellipse rotation in degrees (counter clockwise, default 0.0).

min_points: Minimum number of data points to use. If less amount of points found the grid node considered empty and will be filled with NODATA marker. This is only used if search ellipse is set (both radii are non-zero). Default is 0.

nodata: NODATA marker to fill empty points (default 0.0).

17.5 READING COMMA SEPARATED VALUES

Often you have a text file with a list of comma separated XYZ values to work with (so called CSV file). You can easily use that kind of data source in gdal_grid. All you need is create a virtual dataset header (VRT) for you CSV file and use it as input datasource for gdal_grid. You can find details on VRT format at `Virtual Format` description page.

Here is a small example. Let we have a CSV file called *dem.csv* containing

```
Easting,Northing,Elevation
86943.4,891957,139.13
87124.3,892075,135.01
86962.4,892321,182.04
87077.6,891995,135.01
...
```

For above data we will create *dem.vrt* header with the following content:

```
<OGRVRTDataSource>
    <OGRVRTLayer name="dem">
        <SrcDataSource>dem.csv</SrcDataSource>
        <GeometryType>wkbPoint</GeometryType>
        <GeometryField encoding="PointFromColumns" x="Easting" y="Northing" z="Elevation"/>
    </OGRVRTLayer>
</OGRVRTDataSource>
```

This description specifies so called 2.5D geometry with three coordinates X, Y and Z. Z value will be used for interpolation. Now you can use *dem.vrt* with all OGR programs (start with ogrinfo to test that everything works fine). The datasource will contain single layer called *"dem"* filled with point features constructed from values in CSV file. Using this technique you can handle CSV files with more than three columns, switch columns, etc.

If your CSV file does not contain column headers then it can be handled in the following way:

```
<GeometryField encoding="PointFromColumns" x="field_1" y="field_2" z="field_3"/>
```

`Comma Separated Value` description page contains details on CSV format supported by GDAL/OGR.

17.6 C API

Starting with GDAL 2.1, this utility is also callable from C with GDALGrid().

17.7 EXAMPLE

The following would create raster TIFF file from VRT datasource described in READING COMMA SEPARATED VALUES section using the inverse distance to a power method. Values to interpolate will be read from Z value of geometry record.

```
gdal_grid -a invdist:power=2.0:smoothing=1.0 -txe 85000 89000 -tye 894000 890000 -outsize 400 400 -of GTiff -c
```

The next command does the same thing as the previous one, but reads values to interpolate from the attribute field specified with **-zfield** option instead of geometry record. So in this case X and Y coordinates are being taken from geometry and Z is being taken from the *"Elevation"* field. The GDAL_NUM_THREADS is also set to parallelize the computation.

```
gdal_grid -zfield "Elevation" -a invdist:power=2.0:smoothing=1.0 -txe 85000 89000 -tye 894000 890000 -outsize
```

Chapter 18

gdaldem

Tools to analyze and visualize DEMs.

18.1 SYNOPSIS

```
- 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] [-combined | -multidirectional]
                [-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]

Notes :
  gdaldem generally assumes that x, y and z units are identical.  If x (east-west)
  and y (north-south) units are identical, but z (elevation) units are different, the
  scale (-s) option can be used to set the ratio of vertical units to horizontal.  For
  LatLong projections near the equator, where units of latitude and units of
  longitude are similar, elevation (z) units can be converted to be compatible
  by using scale=370400 (if elevation is in feet) or scale=111120 (if elevation is in
  meters).  For locations not near the equator, it would be best to reproject your
```

```
grid using gdalwarp before using gdaldem.
```

This utility has 7 different modes :

hillshade to generate a shaded relief map from any GDAL-supported elevation raster

slope to generate a slope map from any GDAL-supported elevation raster

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

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

TRI to generate a map of Terrain Ruggedness Index from any GDAL-supported elevation raster

TPI to generate a map of Topographic Position Index from any GDAL-supported elevation raster

roughness to generate a map of roughness from any GDAL-supported elevation raster

The following general options are available :

***input_dem*:** The input DEM raster to be processed

***output_xxx_map*:** The output raster produced

-of *format*: Select the output format. The default is GeoTIFF (GTiff). Use the short format name.

-compute_edges: (GDAL >= 1.8.0) Do the computation at raster edges and near nodata values

-alg *ZevenbergenThorne*: (GDAL >= 1.8.0) Use Zevenbergen & Thorne formula, instead of Horn's formula, to compute slope & aspect. The literature suggests Zevenbergen & Thorne to be more suited to smooth landscapes, whereas Horn's formula to perform better on rougher terrain.

-b *band*: Select an input *band* to be processed. Bands are numbered from 1.

-co *"NAME=VALUE"*: Passes a creation option to the output format driver. Multiple **-co** options may be listed. See `format specific documentation for legal creation options for each format`

-q: Suppress progress monitor and other non-error output.

For all algorithms, except color-relief, a nodata value in the target dataset will be emitted if at least one pixel set to the nodata value is found in the 3x3 window centered around each source pixel. The consequence is that there will be a 1-pixel border around each image set with nodata value. From GDAL 1.8.0, if -compute_edges is specified, gdaldem will compute values at image edges or if a nodata value is found in the 3x3 window, by interpolating missing values.

18.2 Modes

18.2.1 hillshade

This command outputs an 8-bit raster with a nice shaded relief effect. It's very useful for visualizing the terrain. You can optionally specify the azimuth and altitude of the light source, a vertical exaggeration factor and a scaling factor to account for differences between vertical and horizontal units.

The value 0 is used as the output nodata value.

The following specific options are available :

-z *zFactor*: vertical exaggeration used to pre-multiply the elevations

-s *scale*: ratio of vertical units to horizontal. If the horizontal unit of the source DEM is degrees (e.g Lat/Long WGS84 projection), you can use scale=111120 if the vertical units are meters (or scale=370400 if they are in feet)

-az *azimuth*: azimuth of the light, in degrees. 0 if it comes from the top of the raster, 90 from the east, ... The default value, 315, should rarely be changed as it is the value generally used to generate shaded maps.

-alt *altitude*: altitude of the light, in degrees. 90 if the light comes from above the DEM, 0 if it is raking light.

-combined: (starting with GDAL 1.10) combined shading, a combination of slope and oblique shading.

-multidirectional: (starting with GDAL 2.2) multidirectional shading, a combination of hillshading illuminated from 225 deg, 270 deg, 315 deg, and 360 deg azimuth.

Multidirectional hillshading applies the formula of http://pubs.usgs.gov/of/1992/of92-422/of92-422.pdf.

18.2.2 slope

This command will take a DEM raster and output a 32-bit float raster with slope values. You have the option of specifying the type of slope value you want: degrees or percent slope. In cases where the horizontal units differ from the vertical units, you can also supply a scaling factor.

The value -9999 is used as the output nodata value.

The following specific options are available :

-p : if specified, the slope will be expressed as percent slope. Otherwise, it is expressed as degrees

-s *scale*: ratio of vertical units to horizontal. If the horizontal unit of the source DEM is degrees (e.g Lat/Long WGS84 projection), you can use scale=111120 if the vertical units are meters (or scale=370400 if they are in feet)

18.2.3 aspect

This command outputs a 32-bit float raster with values between 0° and 360° representing the azimuth that slopes are facing. The definition of the azimuth is such that : 0° means that the slope is facing the North, 90° it's facing the East, 180° it's facing the South and 270° it's facing the West (provided that the top of your input raster is north oriented). The aspect value -9999 is used as the nodata value to indicate undefined aspect in flat areas with slope=0.

The following specifics options are available :

-trigonometric: return trigonometric angle instead of azimuth. Thus 0° means East, 90° North, 180° West, 270° South

-zero_for_flat: return 0 for flat areas with slope=0, instead of -9999

By using those 2 options, the aspect returned by gdaldem aspect should be identical to the one of GRASS r.slope.aspect. Otherwise, it's identical to the one of Matthew Perry's aspect.cpp utility.

18.2.4 color-relief

This command outputs a 3-band (RGB) or 4-band (RGBA) raster with values are computed from the elevation and a text-based color configuration file, containing the association between various elevation values and the corresponding wished color. By default, the colors between the given elevation values are blended smoothly and the result is a nice colorized DEM. The -exact_color_entry or -nearest_color_entry options can be used to avoid that linear interpolation for values that don't match an index of the color configuration file.

The following specifics options are available :

***color_text_file*:** text-based color configuration file

-alpha : add an alpha channel to the output raster

-exact_color_entry : use strict matching when searching in the color configuration file. If none matching color entry is found, the "0,0,0,0" RGBA quadruplet will be used

-nearest_color_entry : use the RGBA quadruplet corresponding to the closest entry in the color configuration file.

The color-relief mode is the only mode that supports VRT as output format. In that case, it will translate the color configuration file into appropriate LUT elements. Note that elevations specified as percentage will be translated as absolute values, which must be taken into account when the statistics of the source raster differ from the one that was used when building the VRT.

The text-based color configuration file generally contains 4 columns per line : the elevation value and the corresponding Red, Green, Blue component (between 0 and 255). The elevation value can be any floating point value, or the *nv* keyword for the nodata value.. The elevation can also be expressed as a percentage : 0% being the minimum value found in the raster, 100% the maximum value.

An extra column can be optionally added for the alpha component. If it is not specified, full opacity (255) is assumed.

Various field separators are accepted : comma, tabulation, spaces, ':'.

Common colors used by GRASS can also be specified by using their name, instead of the RGB triplet. The supported list is : white, black, red, green, blue, yellow, magenta, cyan, aqua, grey/gray, orange, brown, purple/violet and indigo.

Since GDAL 1.8.0, GMT .cpt palette files are also supported (COLOR_MODEL = RGB only).

Note: the syntax of the color configuration file is derived from the one supported by GRASS r.colors utility. ESRI HDR color table files (.clr) also match that syntax. The alpha component and the support of tab and comma as separators are GDAL specific extensions.

For example :

```
3500   white
2500   235:220:175
50%   190 185 135
700    240 250 150
0      50  180  50
nv     0   0   0   0
```

18.2.5 TRI

This command outputs a single-band raster with values computed from the elevation. TRI stands for Terrain Ruggedness Index, which is defined as the mean difference between a central pixel and its surrounding cells (see Wilson et al 2007, Marine Geodesy 30:3-35).

The value -9999 is used as the output nodata value.

There are no specific options.

18.2.6 TPI

This command outputs a single-band raster with values computed from the elevation. TPI stands for Topographic Position Index, which is defined as the difference between a central pixel and the mean of its surrounding cells (see Wilson et al 2007, Marine Geodesy 30:3-35).

The value -9999 is used as the output nodata value.

There are no specific options.

18.2.7 roughness

This command outputs a single-band raster with values computed from the elevation. Roughness is the largest inter-cell difference of a central pixel and its surrounding cell, as defined in Wilson et al (2007, Marine Geodesy

30:3-35).

The value -9999 is used as the output nodata value.

There are no specific options.

18.3 C API

Starting with GDAL 2.1, this utility is also callable from C with GDALDEMProcessing().

18.4 AUTHORS

Matthew Perry perrygeo@gmail.com, Even Rouault even.rouault@mines-paris.org, Howard Butler hobu.inc@gmail.com, Chris Yesson chris.yesson@ioz.ac.uk

Derived from code by Michael Shapiro, Olga Waupotitsch, Marjorie Larson, Jim Westervelt : U.S. Army CERL, 1993. GRASS 4.1 Reference Manual. U.S. Army Corps of Engineers, Construction Engineering Research Laboratories, Champaign, Illinois, 1-425.

18.5 See also

Documentation of related GRASS utilities :

http://grass.osgeo.org/grass64/manuals/html64_user/r.slope.aspect.html

http://grass.osgeo.org/grass64/manuals/html64_user/r.shaded.relief.html

http://grass.osgeo.org/grass64/manuals/html64_user/r.colors.html

Chapter 19

gdalsrsinfo

Lists info about a given SRS in number of formats (WKT, PROJ.4, etc.)

19.1 SYNOPSIS

```
Usage: gdalsrsinfo [options] srs_def

srs_def may be the filename of a dataset supported by GDAL/OGR from which to extract SRS information
OR any of the usual GDAL/OGR forms (complete WKT, PROJ.4, EPSG:n or a file containing the SRS)

Options:
   [--help-general] [-h]  Show help and exit
   [-p]                   Pretty-print where applicable (e.g. WKT)
   [-V]                   Validate SRS
   [-o out_type]          Output type { default, all, wkt_all, proj4,
                                        wkt, wkt_simple, wkt_noct, wkt_esri,
                                        mapinfo, xml }
```

19.2 DESCRIPTION

The gdalsrsinfo utility reports information about a given SRS from one of the following:

- The filename of a dataset supported by GDAL/OGR which contains SRS information
- Any of the usual GDAL/OGR forms (complete WKT, PROJ.4, EPSG:n or a file containing the SRS)

Output types:

- **default** proj4 and wkt (default option)
- **all** all options available
- **wkt_all** all wkt options available
- **proj4** PROJ.4 string
- **wkt** OGC WKT format (full)
- **wkt_simple** OGC WKT (simplified)
- **wkt_noct** OGC WKT (without OGC CT params)
- **wkt_esri** ESRI WKT format
- **mapinfo** Mapinfo style CoordSys format
- **xml** XML format (GML based)

19.3 EXAMPLE

```
$  gdalsrsinfo   "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"]]


$ gdalsrsinfo -o proj4 osr/data/lcc_esri.prj
'+proj=lcc +lat_1=34.33333333333334 +lat_2=36.16666666666666 +lat_0=33.75 +lon_0=-79 +x_0=609601.22 +y_0=0 +da


$ gdalsrsinfo -o proj4 landsat.tif
PROJ.4 : '+proj=utm +zone=19 +south +datum=WGS84 +units=m +no_defs '


$ gdalsrsinfo  -o wkt -p  "EPSG:32722"

PROJCS["WGS 84 / UTM zone 22S",
    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"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",-51],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",10000000],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32722"]]


$ gdalsrsinfo  -o wkt_all  "EPSG:4618"

OGC WKT :
GEOGCS["SAD69",
    DATUM["South_American_Datum_1969",
        SPHEROID["GRS 1967 Modified",6378160,298.25,
            AUTHORITY["EPSG","7050"]],
        TOWGS84[-57,1,-41,0,0,0,0],
        AUTHORITY["EPSG","6618"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4618"]]

OGC WKT (simple) :
GEOGCS["SAD69",
    DATUM["South_American_Datum_1969",
        SPHEROID["GRS 1967 Modified",6378160,298.25],
        TOWGS84[-57,1,-41,0,0,0,0]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]]
```

```
OGC WKT (no CT) :
GEOGCS["SAD69",
    DATUM["South_American_Datum_1969",
        SPHEROID["GRS 1967 Modified",6378160,298.25]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]]

ESRI WKT :
GEOGCS["SAD69",
    DATUM["D_South_American_1969",
        SPHEROID["GRS_1967_Truncated",6378160,298.25]],
    PRIMEM["Greenwich",0],
    UNIT["Degree",0.0174532925199433295]]
```

Chapter 20

gdalmanage

Identify, delete, rename and copy raster data files.

20.1 SYNOPSIS

```
Usage: gdalmanage mode [-r] [-u] [-f format]
                  datasetname [newdatasetname]
```

20.2 DESCRIPTION

The gdalmanage program can perform various operations on raster data files, depending on the chosen *mode*. This includes identifying raster data types and deleting, renaming or copying the files.

***mode*:** Mode of operation

> **identify *datasetname*:** List data format of file.
>
> **copy *datasetname newdatasetname*:** Create a copy of the raster file with a new name.
>
> **rename *datasetname newdatasetname*:** Change the name of the raster file.
>
> **delete *datasetname*:** Delete raster file.

-r: Recursively scan files/folders for raster files.

-u: Report failures if file type is unidentified.

-f *format*: Specify format of raster file if unknown by the application. Uses short data format name (e.g. *GTiff*).

***datasetname*:** Raster file to operate on.

***newdatasetname*:** For copy and rename modes, you provide a *source* filename and a *target* filename, just like copy and move commands in an operating system.

20.3 EXAMPLES

20.3.1 Using identify mode

Report the data format of the raster file by using the *identify* mode and specifying a data file name:

```
$ gdalmanage identify NE1_50M_SR_W.tif

NE1_50M_SR_W.tif: GTiff
```

Recursive mode will scan subfolders and report the data format:

```
$ gdalmanage identify -r 50m_raster/

NE1_50M_SR_W/ne1_50m.jpg: JPEG
NE1_50M_SR_W/ne1_50m.png: PNG
NE1_50M_SR_W/ne1_50m_20pct.tif: GTiff
NE1_50M_SR_W/ne1_50m_band1.tif: GTiff
NE1_50M_SR_W/ne1_50m_print.png: PNG
NE1_50M_SR_W/NE1_50M_SR_W.aux: HFA
NE1_50M_SR_W/NE1_50M_SR_W.tif: GTiff
NE1_50M_SR_W/ne1_50m_sub.tif: GTiff
NE1_50M_SR_W/ne1_50m_sub2.tif: GTiff
```

20.3.2 Using copy mode

Copy the raster data:

```
$ gdalmanage copy NE1_50M_SR_W.tif ne1_copy.tif
```

20.3.3 Using rename mode

Rename raster data:

```
$ gdalmanage rename NE1_50M_SR_W.tif ne1_rename.tif
```

20.3.4 Using delete mode

Delete the raster data:

```
gdalmanage delete NE1_50M_SR_W.tif
```

Chapter 21

gdal_pansharpen.py

Perform a pansharpen operation.

(Since GDAL 2.1)

21.1 SYNOPSIS

```
gdal_pansharpen [--help-general] pan_dataset {spectral_dataset[,band=num]}+ out_dataset
                [-of format] [-b band]* [-w weight_val]*
                [-r {nearest,bilinear,cubic,cubicspline,lanczos,average}]
                [-threads {ALL_CPUS|number}] [-bitdepth val] [-nodata val]
                [-spat_adjust {union,intersection,none,nonewithoutwarning}]
                [-co NAME=VALUE]* [-q]
```

21.2 DESCRIPTION

The gdal_pansharpen.py script performs a pan-sharpening operation. It can create a "classic" output dataset (such as GeoTIFF), or a VRT dataset describing the pan-sharpening operation.

More details can be found in the VRT tutorial.

-of *format*: Select the output format. The default is GeoTIFF (GTiff). "VRT" can also be used. Use the short format name.

-b *band*: Select band *band* from the input spectral bands for output. Bands are numbered from 1 in the order spectral bands are specified. Multiple **-b** switches may be used. When no -b switch is used, all input spectral bands are set for output.

-w *weight_val*: Specify a weight for the computation of the pseudo panchromatic value. There must be as many -w switches as input spectral bands.

-r *{nearest,bilinear,cubic (default),cubicspline,lanczos,average}*: Select a resampling algorithm.

-threads *{ALL_CPUS,number}*: Specify number of threads to use to do the resampling and pan-sharpening itself. Can be an integer number or ALL_CPUS.

-bitdepth *val*: Specify the bit depth of the panchromatic and spectral bands (e.g. 12). If not specified, the NBITS metadata item from the panchromatic band will be used if it exists.

-nodata *val*: Specify nodata value for bands. Used for the resampling and pan-sharpening computation itself. If not set, deduced from the input bands, provided they have a consistent setting.

-spat_adjust *{union (default),intersection,none,nonewithoutwarning}*: Select behaviour when bands have not the same extent. See *SpatialExtentAdjustment* documentation in VRT tutorial

-co *"NAME=VALUE"*: Passes a creation option to the output format driver. Multiple **-co** options may be listed. See format specific documentation for legal creation options for each format.

-q: Suppress progress monitor and other non-error output.

pan_dataset Dataset with panchromatic band (first band will be used).

spectral_dataset[,band=num] Dataset with one or several spectral bands. If the band option is not specified, all bands of the datasets are taken into account. Otherwise, only the specified (num)th band. The same dataset can be repeated several times.

out_dataset Output dataset

Bands should be in the same projection.

21.3 EXAMPLE

With spectral bands in a single dataset :

```
gdal_pansharpen.py panchro.tif multispectral.tif pansharpened_out.tif
```

With a few spectral bands from a single dataset, reordered :

```
gdal_pansharpen.py panchro.tif multispectral.tif,band=3 multispectral.tif,band=2 multispectral.tif,band=1 pans
```

With spectral bands in several datasets :

```
gdal_pansharpen.py panchro.tif band1.tif band2.tif band3.tif pansharpened_out.tif
```

Specify weights:

```
gdal_pansharpen.py -w 0.7 -w 0.2 -w 0.1 panchro.tif multispectral.tif pansharpened_out.tif
```

Specify RGB bands from a RGBNir multispectral dataset while computing the pseudo panchromatic intensity on the 4 RGBNir bands:

```
gdal_pansharpen.py -b 1 -b 2 -b 3 panchro.tif rgbnir.tif pansharpened_out.tif
```

Chapter 22

gdallocationinfo

raster query tool

22.1 SYNOPSIS

```
Usage: gdallocationinfo [--help-general] [-xml] [-lifonly] [-valonly]
                        [-b band]* [-overview overview_level]
                        [-l_srs srs_def] [-geoloc] [-wgs84]
                        [-oo NAME=VALUE]* srcfile [x y]
```

22.2 DESCRIPTION

The gdallocationinfo utility provide a mechanism to query information about a pixel given its location in one of a variety of coordinate systems. Several reporting options are provided.

-xml: The output report will be XML formatted for convenient post processing.

-lifonly: The only output is filenames production from the LocationInfo request against the database (i.e. for identifying impacted file from VRT).

-valonly: The only output is the pixel values of the selected pixel on each of the selected bands.

-b *band*: Selects a band to query. Multiple bands can be listed. By default all bands are queried.

-overview *overview_level*: Query the (overview_level)th overview (overview_level=1 is the 1st overview), instead of the base band. Note that the x,y location (if the coordinate system is pixel/line) must still be given with respect to the base band.

-l_srs *srs def*: The coordinate system of the input x, y location.

-geoloc: Indicates input x,y points are in the georeferencing system of the image.

-wgs84: Indicates input x,y points are WGS84 long, lat.

-oo *NAME=VALUE*: (starting with GDAL 2.0) Dataset open option (format specific)

***srcfile*:** The source GDAL raster datasource name.

***x*:** X location of target pixel. By default the coordinate system is pixel/line unless -l_srs, -wgs84 or -geoloc supplied.

***y*:** Y location of target pixel. By default the coordinate system is pixel/line unless -l_srs, -wgs84 or -geoloc supplied.

This utility is intended to provide a variety of information about a pixel. Currently it reports three things:

- The location of the pixel in pixel/line space.
- The result of a LocationInfo metadata query against the datasource - currently this is only implemented for VRT files which will report the file(s) used to satisfy requests for that pixel.
- The raster pixel value of that pixel for all or a subset of the bands.
- The unscaled pixel value if a Scale and/or Offset apply to the band.

The pixel selected is requested by x/y coordinate on the command line, or read from stdin. More than one coordinate pair can be supplied when reading coordinates from stdin. By default pixel/line coordinates are expected. However with use of the -geoloc, -wgs84, or -l_srs switches it is possible to specify the location in other coordinate systems.

The default report is in a human readable text format. It is possible to instead request xml output with the -xml switch.

For scripting purposes, the -valonly and -lifonly switches are provided to restrict output to the actual pixel values, or the LocationInfo files identified for the pixel.

It is anticipated that additional reporting capabilities will be added to gdallocationinfo in the future.

22.3 EXAMPLE

Simple example reporting on pixel (256,256) on the file utm.tif.

```
$ gdallocationinfo utm.tif 256 256
Report:
  Location: (256P,256L)
  Band 1:
    Value: 115
```

Query a VRT file providing the location in WGS84, and getting the result in xml.

```
$ gdallocationinfo -xml -wgs84 utm.vrt -117.5 33.75
<Report pixel="217" line="282">
  <BandReport band="1">
    <LocationInfo>
      <File>utm.tif</File>
    </LocationInfo>
    <Value>16</Value>
  </BandReport>
</Report>
```

Chapter 23

gdalwarp

image reprojection and warping utility

23.1 SYNOPSIS

```
gdalwarp [--help-general] [--formats]
    [-s_srs srs_def] [-t_srs srs_def] [-to "NAME=VALUE"] [-novshiftgrid]
    [-order n | -tps | -rpc | -geoloc] [-et err_threshold]
    [-refine_gcps tolerance [minimum_gcps]]
    [-te xmin ymin xmax ymax] [-te_srs srs_def]
    [-tr xres yres] [-tap] [-ts width height]
    [-ovr level|AUTO|AUTO-n|NONE] [-wo "NAME=VALUE"] [-ot Byte/Int16/...] [-wt Byte/Int16]
    [-srcnodata "value [value...]"] [-dstnodata "value [value...]"]
    [-srcalpha|-nosrcalpha] [-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]
    [-nomd] [-cvmd meta_conflict_value] [-setci] [-oo NAME=VALUE]*
    [-doo NAME=VALUE]*
    srcfile* dstfile
```

23.2 DESCRIPTION

The gdalwarp utility is an image mosaicing, reprojection and warping utility. The program can reproject to any supported projection, and can also apply GCPs stored with the image if the image is "raw" with control information.

-s_srs *srs def*: source spatial reference set. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG PCS and GCSes (i.e. EPSG-:4296), PROJ.4 declarations (as above), or the name of a .prj file containing well known text. Starting with GDAL 2.2, if the SRS has an explicit vertical datum that points to a PROJ.4 geoidgrids, and the input dataset is a single band dataset, a vertical correction will be applied to the values of the dataset.

-t_srs *srs_def*: target spatial reference set. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG PCS and GCSes (i.e. EPSG-:4296), PROJ.4 declarations (as above), or the name of a .prj file containing well known text. Starting with GDAL 2.2, if the SRS has an explicit vertical datum that points to a PROJ.4 geoidgrids, and the input dataset is a single band dataset, a vertical correction will be applied to the values of the dataset.

-to *NAME=VALUE*: set a transformer option suitable to pass to GDALCreateGenImgProjTransformer2().

-novshiftgrid (GDAL >= 2.2) Disable the use of vertical datum shift grids when one of the source or target SRS has an explicit vertical datum, and the input dataset is a single band dataset.

-order *n*: order of polynomial used for warping (1 to 3). The default is to select a polynomial order based on the number of GCPs.

-tps: Force use of thin plate spline transformer based on available GCPs.

-rpc: Force use of RPCs.

-geoloc: Force use of Geolocation Arrays.

-et *err_threshold*: error threshold for transformation approximation (in pixel units - defaults to 0.125, unless, starting with GDAL 2.1, the RPC_DEM warping option is specified, in which case, an exact transformer, i.e. err_threshold=0, will be used).

-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.

-te *xmin ymin xmax ymax*: set georeferenced extents of output file to be created (in target SRS by default, or in the SRS specified with -te_srs)

-te_srs *srs_def*: (GDAL >= 2.0) Specifies the SRS in which to interpret the coordinates given with -te. The *srs_def* may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT. This must not be confused with -t_srs which is the target SRS of the output dataset. -te_srs is a convenience e.g. when knowing the output coordinates in a geodetic long/lat SRS, but still wanting a result in a projected coordinate system.

-tr *xres yres*: set output file resolution (in target georeferenced units)

-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*: set output file size in pixels and lines. If width or height is set to 0, the other dimension will be guessed from the computed resolution. Note that -ts cannot be used with -tr

-ovr *level|AUTO|AUTO-n|NONE*>: (GDAL >= 2.0) To specify which overview level of source files must be used. The default choice, AUTO, will select the overview level whose resolution is the closest to the target resolution. Specify an integer value (0-based, i.e. 0=1st overview level) to select a particular level. Specify AUTO-n where n is an integer greater or equal to 1, to select an overview level below the AUTO one. Or specify NONE to force the base resolution to be used (can be useful if overviews have been generated with a low quality resampling method, and the warping is done using a higher quality resampling method).

-wo *"NAME=VALUE"*: Set a warp option. The GDALWarpOptions::papszWarpOptions docs show all options. Multiple **-wo** options may be listed.

-ot *type*: For the output bands to be of the indicated data type.

-wt *type*: Working pixel data type. The data type of pixels in the source image and destination image buffers.

-r *resampling_method*: Resampling method to use. Available methods are:

near: nearest neighbour resampling (default, fastest algorithm, worst interpolation quality).

bilinear: bilinear resampling.

cubic: cubic resampling.

cubicspline: cubic spline resampling.

lanczos: Lanczos windowed sinc resampling.

average: average resampling, computes the average of all non-NODATA contributing pixels. (GDAL >= 1.10.0)

mode: mode resampling, selects the value which appears most often of all the sampled points. (GDAL >= 1.10.0)

max: maximum resampling, selects the maximum value from all non-NODATA contributing pixels. (GDAL >= 2.0.0)

min: minimum resampling, selects the minimum value from all non-NODATA contributing pixels. (GDAL >= 2.0.0)

med: median resampling, selects the median value of all non-NODATA contributing pixels. (GDAL >= 2.0.0)

q1: first quartile resampling, selects the first quartile value of all non-NODATA contributing pixels. (GDAL >= 2.0.0)

q3: third quartile resampling, selects the third quartile value of all non-NODATA contributing pixels. (GDAL >= 2.0.0)

-srcnodata *value [value...]*: Set nodata masking values for input bands (different values can be supplied for each band). If more than one value is supplied all values should be quoted to keep them together as a single operating system argument. Masked values will not be used in interpolation. Use a value of `None` to ignore intrinsic nodata settings on the source dataset.

-dstnodata *value [value...]*: Set nodata values for output bands (different values can be supplied for each band). If more than one value is supplied all values should be quoted to keep them together as a single operating system argument. New files will be initialized to this value and if possible the nodata value will be recorded in the output file. Use a value of `None` to ensure that nodata is not defined (GDAL>=1.11). If this argument is not used then nodata values will be copied from the source dataset (GDAL>=1.11).

-srcalpha: Force the last band of a source image to be considered as a source alpha band.

-nosrcalpha: Prevent the alpha band of a source image to be considered as such (it will be warped as a regular band) (GDAL>=2.2).

-dstalpha: Create an output alpha band to identify nodata (unset/transparent) pixels.

-wm *memory_in_mb*: Set the amount of memory (in megabytes) that the warp API is allowed to use for caching.

-multi: Use multithreaded warping implementation. Multiple threads will be used to process chunks of image and perform input/output operation simultaneously.

-q: Be quiet.

-of *format*: Select the output format. The default is GeoTIFF (GTiff). Use the short format name.

-co *"NAME=VALUE"*: passes a creation option to the output format driver. Multiple **-co** options may be listed. See `format specific documentation for legal creation options for each format`

-cutline *datasource*: Enable use of a blend cutline from the name OGR support datasource.

-cl *layername*: Select the named layer from the cutline datasource.

-cwhere *expression*: Restrict desired cutline features based on attribute query.

-csql *query*: Select cutline features using an SQL query instead of from a layer with -cl.

-cblend *distance*: Set a blend distance to use to blend over cutlines (in pixels).

-crop_to_cutline: (GDAL >= 1.8.0) Crop the extent of the target dataset to the extent of the cutline.

-overwrite: (GDAL >= 1.8.0) Overwrite the target dataset if it already exists.

-nomd: (GDAL >= 1.10.0) Do not copy metadata. Without this option, dataset and band metadata (as well as some band information) will be copied from the first source dataset. Items that differ between source datasets will be set to * (see -cvmd option).

-cvmd *meta_conflict_value*: (GDAL >= 1.10.0) Value to set metadata items that conflict between source datasets (default is "*"). Use "" to remove conflicting items.

-setci: (GDAL >= 1.10.0) Set the color interpretation of the bands of the target dataset from the source dataset.

-oo *NAME=VALUE*: (starting with GDAL 2.0) Dataset open option (format specific)

-doo *NAME=VALUE*: (starting with GDAL 2.1) Output dataset open option (format specific)

***srcfile*:** The source file name(s).

***dstfile*:** The destination file name.

Mosaicing into an existing output file is supported if the output file already exists. The spatial extent of the existing file will not be modified to accommodate new data, so you may have to remove it in that case, or use the -overwrite option.

Polygon cutlines may be used as a mask to restrict the area of the destination file that may be updated, including blending. If the OGR layer containing the cutline features has no explicit SRS, the cutline features must be in the SRS of the destination file. When writing to a not yet existing target dataset, its extent will be the one of the original raster unless -te or -crop_to_cutline are specified.

When doing vertical shift adjustments, the transformer option -to ERROR_ON_MISSING_VERT_SHIFT=YES can be used to error out as soon as a vertical shift value is missing (instead of 0 being used).

23.3 EXAMPLES

- For instance, an eight bit spot scene stored in GeoTIFF with control points mapping the corners to lat/long could be warped to a UTM projection with a command like this:

```
gdalwarp -t_srs '+proj=utm +zone=11 +datum=WGS84' -overwrite raw_spot.tif utm11.tif
```

- For instance, the second channel of an ASTER image stored in HDF with control points mapping the corners to lat/long could be warped to a UTM projection with a command like this:

```
gdalwarp -overwrite HDF4_SDS:ASTER_L1B:"pg-PR1B0000-2002031402_100_001":2 pg-PR1B0000-2002031402_100_001_2.tif
```

- (GDAL >= 2.2) To apply a cutline on a un-georeferenced image and clip from pixel (220,60) to pixel (1160,690):

```
gdalwarp -overwrite -to SRC_METHOD=NO_GEOTRANSFORM -to DST_METHOD=NO_GEOTRANSFORM -te 220 60 1160 690 -cutline
```

where cutline.csv content is like:

```
id,WKT
1,"POLYGON((....))"
```

- (GDAL >= 2.2) To transform a DEM from geoid elevations (using EGM96) to WGS84 ellipsoidal heights:

```
gdalwarp -override in_dem.tif out_dem.tif -s_srs EPSG:4326+5773 -t_srs EPSG:4979
```

23.4 SEE ALSO

Wiki page discussing options and behaviours of gdalwarp

Chapter 24

GNM Utility Programs

There are several Geographical Network Model (GNM) utilities:

- gnmmanage - Manages networks
- gnmanalyse - Analyses networks

Chapter 25

gnmmanage

Manages networks

25.1 SYNOPSIS

```
gnmmanage [--help][-q][-quiet][--long-usage]
          [info]
          [create [-f format_name] [-t_srs srs_name] [-dsco NAME=VALUE]... ]
          [import src_dataset_name] [-l layer_name]
          [connect gfid_src gfid_tgt gfid_con [-c cost] [-ic inv_cost] [-dir dir]]
          [disconnect gfid_src gfid_tgt gfid_con]
          [rule rule_str]
          [autoconnect tolerance]
          [delete]
          [change [-bl gfid][-unbl gfid][-unblall]]
          gnm_name [layer [layer ...]]
```

25.2 DESCRIPTION

The gnmmanage program can perform various managing operations on geographical networks in GDAL. In addition to creating and deleting networks this includes capabilities of managing network's features, topology and rules.

info: Different information about network: system and class layers, network metadata, network spatial reference.

create: Create network.

-f *format_name*: Output file format name.

-t_srs *srs_name*: Spatial reference input.

-dsco *NAME=VALUE*: Network creation option set as pair name=value.

import *src_dataset_name*: Import layer with dataset name to copy.

-l *layer_name*: Layer name in dataset. If unset, 0 layer is copied.

connect *gfid_src gfid_tgt gfid_con*: Make a topological connection, where the gfid_src and gfid_tgt are vertexes and gfid_con is edge (gfid_con can be -1, so the system edge will be inserted).

-c *cost* -ic *inv_cost* -dir *dir*: Manually assign the following values: the cost (weight), inverse cost and direction of the edge (optional).

disconnect *gfid_src gfid_tgt gfid_con*: Removes the connection from the graph.

rule *rule_str*: Creates a rule in the network by the given rule_str string.

autoconnect *tolerance*: Create topology automatically with the given double tolerance and layer names. In no layer name provided all layers of network will be used.

delete: Delete network.

change: Change blocking state of network edges or vertices.

-bl *gfid*: Block feature before the main operation. Blocking features are saved in the special layer.

-unbl *gfid*: Unblock feature before the main operation.

-unblall: Unblock all blocked features before the main operation.

***gnm_name*:** The network to work with (path and name).

***layer*:** The network layer name.

Chapter 26

gnmanalyse

Analyses networks

26.1 SYNOPSIS

```
gnmanalyse [--help][-q][-quiet][--long-usage]
           [dijkstra start_gfid end_gfid [[-alo NAME=VALUE] ...]]]
           [kpaths start_gfid end_gfid k [[-alo NAME=VALUE] ...]]]
           [resource [[-alo NAME=VALUE] ...]]]
           [-ds ds_name][-f ds_format][-l layer_name]
           [[-dsco NAME=VALUE] ...][-lco NAME=VALUE]
           gnm_name
```

26.2 DESCRIPTION

The gnmanalyse program provides analysing capabilities of geographical networks in GDAL. The results of calculations are return in an OGRLayer format or as a console text output if such layer is undefined. All calculations are made considering the blocking state of features.

dijkstra *start_gfid end_gfid*: Calculates the best path between two points using Dijkstra algorithm from start_gfid point to end_gfid point.

kpaths *start_gfid end_gfid*: Calculates K shortest paths between two points using Yen's algorithm (which internally uses Dijkstra algorithm for single path calculating) from start_gfid point to end_gfid point.

resource: Calculates the "resource distribution". The connected components search is performed using breadth-first search and starting from that features which are marked by rules as 'EMITTERS'.

-ds *ds_name*: The name&path of the dataset to save the layer with resulting paths. Not need to be existed dataset.

-f *ds_format*: Define this to set the format of newly created dataset.

-l *layer_name*: The name of the resulting layer. If the layer exist already - it will be rewritten.

***gnm_name*:** The network to work with (path and name).

-dsco *NAME=VALUE*: Dataset creation option (format specific)

-lco*NAME=VALUE*: Layer creation option (format specific)

-alo*NAME=VALUE*: Algorithm option (format specific)

Chapter 27

OGR Utility Programs

A collection of OGR related programs.

The following utilities are distributed as part of the OGR Simple Features toolkit:

- ogrinfo - Lists information about an OGR supported data source
- ogr2ogr - Converts simple features data between file formats
- ogrtindex - Creates a tileindex
- ogrlineref - Create linear reference and provide some calculations using it
- ogrmerge - Merge several vector datasets into a single one

27.1 General Command Line Switches

All GDAL OGR command line utility programs support the following "general" options.

--version Report the version of GDAL and exit.

--formats List all vector formats supported by this GDAL build (read-only and read-write) and exit. The format support is indicated as follows: 'ro' is read-only driver; 'rw' is read or write (i.e. supports CreateCopy); 'rw+' is read, write and update (i.e. supports Create). A 'v' is appended for formats supporting virtual IO (/vsimem, /vsigzip, /vsizip, etc). A 's' is appended for formats supporting subdatasets.

--format *format* List detailed information about a single format driver. The *format* should be the short name reported in the **--formats** list, such as GML.

--optfile *file* Read the named file and substitute the contents into the command line options list. Lines beginning with # will be ignored. Multi-word arguments may be kept together with double quotes.

--config *key value* Sets the named `configuration keyword` to the given value, as opposed to setting them as environment variables. Some common configuration keywords are SHAPE_ENCODING (force shapefile driver to read DBF files with the given character encoding) and CPL_TEMPDIR (define the location of temporary files). Individual drivers may be influenced by other configuration options.

--debug *value* Control what debugging messages are emitted. A value of *ON* will enable all debug messages. A value of *OFF* will disable all debug messages. Another value will select only debug messages containing that string in the debug prefix code.

--help-general Gives a brief usage message for the generic GDAL OGR command line options and exit.

Chapter 28

ogrinfo

Lists information about an OGR supported data source.

28.1 SYNOPSIS

```
ogrinfo [--help-general] [-ro] [-q] [-where restricted_where|\@filename]
        [-spat xmin ymin xmax ymax] [-geomfield field] [-fid fid]
        [-sql statement|\@filename] [-dialect dialect] [-al] [-rl] [-so] [-fields={YES/NO}]
        [-geom={YES/NO/SUMMARY/WKT/ISO_WKT}] [-formats] [[-oo NAME=VALUE] ...]
        [-nomd] [-listmdd] [-mdd domain|`all`]*
        [-nocount] [-noextent]
        datasource_name [layer [layer ...]]
```

28.2 DESCRIPTION

The ogrinfo program lists various information about an OGR supported data source to stdout (the terminal).

-ro: Open the data source in read-only mode.

-al: List all features of all layers (used instead of having to give layer names as arguments).

-rl: (Available in GDAL 2.2) Enable random layer reading mode, i.e. iterate over features in the order they are found in the dataset, and not layer per layer. This can be significantly faster for some formats (for example OSM, GMLAS).

-so: Summary Only: suppress listing of features, show only the summary information like projection, schema, feature count and extents.

-q: Quiet verbose reporting of various information, including coordinate system, layer schema, extents, and feature count.

-where *restricted_where*: An attribute query in a restricted form of the queries used in the SQL WHERE statement. Only features matching the attribute query will be reported. Starting with GDAL 2.1, the \filename syntax can be used to indicate that the content is in the pointed filename.

-sql *statement*: Execute the indicated SQL statement and return the result. Starting with GDAL 2.1, the @filename syntax can be used to indicate that the content is in the pointed filename.

-dialect *dialect*: SQL dialect. In some cases can be used to use (unoptimized) OGR SQL instead of the native SQL of an RDBMS by passing OGRSQL. Starting with GDAL 1.10, the "SQLITE" dialect can also be used with any datasource.

-spat *xmin ymin xmax ymax*: The area of interest. Only features within the rectangle will be reported.

-geomfield *field*: (OGR >= 1.11) Name of the geometry field on which the spatial filter operates on.

-fid *fid*: If provided, only the feature with this feature id will be reported. Operates exclusive of the spatial or attribute queries. Note: if you want to select several features based on their feature id, you can also use the fact the 'fid' is a special field recognized by OGR SQL. So, '-where "fid in (1,3,5)"' would select features 1, 3 and 5.

-fields={YES/NO}: (starting with GDAL 1.6.0) If set to NO, the feature dump will not display field values. Default value is YES.

-geom={YES/NO/SUMMARY/WKT/ISO_WKT}: (starting with GDAL 1.6.0) If set to NO, the feature dump will not display the geometry. If set to SUMMARY, only a summary of the geometry will be displayed. If set to YES or ISO_WKT, the geometry will be reported in full OGC WKT format. If set to WKT the geometry will be reported in legacy WKT. Default value is YES. (WKT and ISO_WKT are available starting with GDAL 2.1, which also changes the default to ISO_WKT)

-oo *NAME=VALUE*: (starting with GDAL 2.0) Dataset open option (format specific)

-nomd (starting with GDAL 2.0) Suppress metadata printing. Some datasets may contain a lot of metadata strings.

-listmdd (starting with GDAL 2.0) List all metadata domains available for the dataset.

-mdd domain (starting with GDAL 2.0) Report metadata for the specified domain. "all" can be used to report metadata in all domains

-nocount (starting with GDAL 2.0) Suppress feature count printing.

-noextent (starting with GDAL 2.0) Suppress spatial extent printing.

--formats: List the format drivers that are enabled.

***datasource_name*:** The data source to open. May be a filename, directory or other virtual name. See the OGR Vector Formats list for supported datasources.

***layer*:** One or more layer names may be reported.

If no layer names are passed then ogrinfo will report a list of available layers (and their layer wide geometry type). If layer name(s) are given then their extents, coordinate system, feature count, geometry type, schema and all features matching query parameters will be reported to the terminal. If no query parameters are provided, all features are reported.

Geometries are reported in OGC WKT format.

28.3 EXAMPLE

Example reporting all layers in an NTF file:

```
% ogrinfo wrk/SHETLAND_ISLANDS.NTF
INFO: Open of 'wrk/SHETLAND_ISLANDS.NTF'
using driver 'UK .NTF' successful.
1: BL2000_LINK (Line String)
2: BL2000_POLY (None)
3: BL2000_COLLECTIONS (None)
4: FEATURE_CLASSES (None)
```

Example using an attribute query is used to restrict the output of the features in a layer:

```
% ogrinfo -ro -where 'GLOBAL_LINK_ID=185878' wrk/SHETLAND_ISLANDS.NTF BL2000_LINK
INFO: Open of 'wrk/SHETLAND_ISLANDS.NTF'
using driver 'UK .NTF' successful.

Layer name: BL2000_LINK
Geometry: Line String
Feature Count: 1
Extent: (419794.100000, 1069031.000000) - (419927.900000, 1069153.500000)
Layer SRS WKT:
```

```
PROJCS["OSGB 1936 / British National Grid",
    GEOGCS["OSGB 1936",
        DATUM["OSGB_1936",
            SPHEROID["Airy 1830",6377563.396,299.3249646]],
        PRIMEM["Greenwich",0],
        UNIT["degree",0.0174532925199433]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",49],
    PARAMETER["central_meridian",-2],
    PARAMETER["scale_factor",0.999601272],
    PARAMETER["false_easting",400000],
    PARAMETER["false_northing",-100000],
    UNIT["metre",1]]
LINE_ID: Integer (6.0)
GEOM_ID: Integer (6.0)
FEAT_CODE: String (4.0)
GLOBAL_LINK_ID: Integer (10.0)
TILE_REF: String (10.0)
OGRFeature(BL2000_LINK):2
  LINE_ID (Integer) = 2
  GEOM_ID (Integer) = 2
  FEAT_CODE (String) = (null)
  GLOBAL_LINK_ID (Integer) = 185878
  TILE_REF (String) = SHETLAND I
  LINESTRING (419832.100 1069046.300,419820.100 1069043.800,419808.300
  1069048.800,419805.100 1069046.000,419805.000 1069040.600,419809.400
  1069037.400,419827.400 1069035.600,419842 1069031,419859.000
  1069032.800,419879.500 1069049.500,419886.700 1069061.400,419890.100
  1069070.500,419890.900 1069081.800,419896.500 1069086.800,419898.400
  1069092.900,419896.700 1069094.800,419892.500 1069094.300,419878.100
  1069085.600,419875.400 1069087.300,419875.100 1069091.100,419872.200
  1069094.600,419890.400 1069106.400,419907.600 1069112.800,419924.600
  1069133.800,419927.900 1069146.300,419927.600 1069152.400,419922.600
  1069153.500,419917.100 1069153.500,419911.500 1069153.000,419908.700
  1069152.500,419903.400 1069150.800,419898.800 1069149.400,419894.800
  1069149.300,419890.700 1069149.400,419890.600 1069149.400,419880.800
  1069149.800,419876.900 1069148.900,419873.100 1069147.500,419870.200
  1069146.400,419862.100 1069143.000,419860 1069142,419854.900
  1069138.600,419850 1069135,419848.800 1069134.100,419843
  1069130,419836.200 1069127.600,419824.600 1069123.800,419820.200
  1069126.900,419815.500 1069126.900,419808.200 1069116.500,419798.700
  1069117.600,419794.100 1069115.100,419796.300 1069109.100,419801.800
  1069106.800,419805.000  1069107.300)
```

Chapter 29

ogr2ogr

Converts simple features data between file formats.

29.1 SYNOPSIS

```
Usage: ogr2ogr [--help-general] [-skipfailures] [-append] [-update]
               [-select field_list] [-where restricted_where|\@filename]
               [-progress] [-sql <sql statement>|\@filename] [-dialect dialect]
               [-preserve_fid] [-fid FID] [-limit nb_features]
               [-spat xmin ymin xmax ymax] [-spat_srs srs_def] [-geomfield field]
               [-a_srs srs_def] [-t_srs srs_def] [-s_srs srs_def]
               [-f format_name] [-overwrite] [[-dsco NAME=VALUE] ...]
               dst_datasource_name src_datasource_name
               [-lco NAME=VALUE] [-nln name]
               [-nlt type|PROMOTE_TO_MULTI|CONVERT_TO_LINEAR|CONVERT_TO_CURVE]
               [-dim XY|XYZ|XYM|XYZM|2|3|layer_dim] [layer [layer ...]]

Advanced options :
               [-gt n]
               [[-oo NAME=VALUE] ...] [[-doo NAME=VALUE] ...]
               [-clipsrc [xmin ymin xmax ymax]|WKT|datasource|spat_extent]
               [-clipsrcsql sql_statement] [-clipsrclayer layer]
               [-clipsrcwhere expression]
               [-clipdst [xmin ymin xmax ymax]|WKT|datasource]
               [-clipdstsql sql_statement] [-clipdstlayer layer]
               [-clipdstwhere expression]
               [-wrapdateline] [-datelineoffset val]
               [[-simplify tolerance] | [-segmentize max_dist]]
               [-addfields] [-unsetFid]
               [-relaxedFieldNameMatch] [-forceNullable] [-unsetDefault]
               [-fieldTypeToString All|(type1[,type2]*)] [-unsetFieldWidth]
               [-mapFieldType type1|All=type2[,type3=type4]*]
               [-fieldmap identity | index1[,index2]*]
               [-splitlistfields] [-maxsubfields val]
               [-explodecollections] [-zfield field_name]
               [-gcp pixel line easting northing [elevation]]* [-order n | -tps]
               [-nomd] [-mo "META-TAG=VALUE"]* [-noNativeData]
```

29.2 DESCRIPTION

This program can be used to convert simple features data between file formats performing various operations during the process such as spatial or attribute selections, reducing the set of attributes, setting the output coordinate system or even reprojecting the features during translation.

-f*format_name*: output file format name (default is ESRI Shapefile), some possible values are:

```
-f "ESRI Shapefile"
```

```
-f "TIGER"
-f "MapInfo File"
-f "GML"
-f "PostgreSQL"
```

-append: Append to existing layer instead of creating new

-overwrite: Delete the output layer and recreate it empty

-update: Open existing output datasource in update mode rather than trying to create a new one

-select*field_list*: Comma-delimited list of fields from input layer to copy to the new layer. A field is skipped if mentioned previously in the list even if the input layer has duplicate field names. (Defaults to all; any field is skipped if a subsequent field with same name is found.) Starting with OGR 1.11, geometry fields can also be specified in the list.

-progress: (starting with GDAL 1.7.0) Display progress on terminal. Only works if input layers have the "fast feature count" capability.

-sql *sql_statement*: SQL statement to execute. The resulting table/layer will be saved to the output. Starting with GDAL 2.1, the @filename syntax can be used to indicate that the content is in the pointed filename.

-dialect *dialect*: SQL dialect. In some cases can be used to use (unoptimized) OGR SQL instead of the native SQL of an RDBMS by passing OGRSQL. Starting with GDAL 1.10, the "SQLITE" dialect can also be used with any datasource.

-where*restricted_where*: Attribute query (like SQL WHERE). Starting with GDAL 2.1, the @filename syntax can be used to indicate that the content is in the pointed filename.

-skipfailures: Continue after a failure, skipping the failed feature.

-spat*xmin ymin xmax ymax*: spatial query extents, in the SRS of the source layer(s) (or the one specified with -spat_srs). Only features whose geometry intersects the extents will be selected. The geometries will not be clipped unless -clipsrc is specified

-spat_srs*srs_def*: (OGR >= 2.0) Override spatial filter SRS.

-geomfield *field*: (OGR >= 1.11) Name of the geometry field on which the spatial filter operates on.

-dsco *NAME=VALUE*: Dataset creation option (format specific)

-lco*NAME=VALUE*: Layer creation option (format specific)

-nln*name*: Assign an alternate name to the new layer

-nlt*type*: Define the geometry type for the created layer. One of NONE, GEOMETRY, POINT, LINESTRING, POLYGON, GEOMETRYCOLLECTION, MULTIPOINT, MULTIPOLYGON or MULTILINESTRING. And CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON, MULTICURVE and MULTISURFACE for GDAL 2.0 non-linear geometry types. Add "Z", "M", or "ZM" to the name to get coordinates with elevation, measure, or elevation and measure. Starting with GDAL 1.10, PROMOTE_TO_MULTI can be used to automatically promote layers that mix polygon or multipolygons to multipolygons, and layers that mix linestrings or multilinestrings to multilinestrings. Can be useful when converting shapefiles to PostGIS and other target drivers that implement strict checks for geometry types. Starting with GDAL 2.0, CONVERT_TO_LINEAR can be used to to convert non-linear geometries types into linear geometries by approximating them, and CONVERT_TO_CURVE to promote a non-linear type to its generalized curve type (POLYGON to CURVEPOLYGON, MULTIPOLYGON to MULTISURFACE, LINESTRING to COMPOUNDCURVE, MULTILINESTRING to MULTICURVE). Starting with 2.1 the type can be defined as measured ("25D" remains as an alias for single "Z").

-dim*val*: (starting with GDAL 1.10) Force the coordinate dimension to val (valid values are XY, XYZ, XYM, and XYZM - for backwards compatibility 2 is an alias for XY and 3 is an alias for XYZ). This affects both the layer geometry type, and feature geometries. Starting with GDAL 1.11, the value can be set to "layer_dim" to instruct feature geometries to be promoted to the coordinate dimension declared by the layer. Support for M was added in GDAL 2.1

-a_srs*srs_def*: Assign an output SRS

-t_srs*srs_def*: Reproject/transform to this SRS on output

-s_srs*srs_def*: Override source SRS

-preserve_fid: Use the FID of the source features instead of letting the output driver to automatically assign a new one. Note: starting with GDAL 2.0, if not in append mode, this behaviour becomes the default if the output driver has a FID layer creation option. In which case the name of the source FID column will be used and source feature IDs will be attempted to be preserved. This behaviour can be disabled by setting -unsetFid

-fid *fid*: If provided, only the feature with this feature id will be reported. Operates exclusive of the spatial or attribute queries. Note: if you want to select several features based on their feature id, you can also use the fact the 'fid' is a special field recognized by OGR SQL. So, '-where "fid in (1,3,5)"' would select features 1, 3 and 5.

-limit *nb_features*: (starting with GDAL 2.2): to limit the number of features, per layer.

Srs_def can be a full WKT definition (hard to escape properly), or a well known definition (i.e. EPSG:4326) or a file with a WKT definition.

Advanced options :

-oo *NAME=VALUE*: (starting with GDAL 2.0) Input dataset open option (format specific)

-doo *NAME=VALUE*: (starting with GDAL 2.0) Destination dataset open option (format specific), only valid in -update mode

-gt *n*: group *n* features per transaction (default 20000 in OGR 1.11, 200 in previous releases). Increase the value for better performance when writing into DBMS drivers that have transaction support. Starting with GDAL 2.0, n can be set to unlimited to load the data into a single transaction.

-ds_transaction: (starting with GDAL 2.0) Force the use of a dataset level transaction (for drivers that support such mechanism), especially for drivers such as FileGDB that only support dataset level transaction in emulation mode.

-clipsrc*[xmin ymin xmax ymax]|WKT|datasource|spat_extent*: (starting with GDAL 1.7.0) clip geometries to the specified bounding box (expressed in source SRS), WKT geometry (POLYGON or MULTIPOLYGON), from a datasource or to the spatial extent of the **-spat** option if you use the *spat_extent* keyword. When specifying a datasource, you will generally want to use it in combination of the **-clipsrclayer**, **-clipsrcwhere** or **-clipsrcsql** options

-clipsrcsql *sql_statement*: Select desired geometries using an SQL query instead.

-clipsrclayer *layername*: Select the named layer from the source clip datasource.

-clipsrcwhere *expression*: Restrict desired geometries based on attribute query.

-clipdst*xmin ymin xmax ymax*: (starting with GDAL 1.7.0) clip geometries after reprojection to the specified bounding box (expressed in dest SRS), WKT geometry (POLYGON or MULTIPOLYGON) or from a datasource. When specifying a datasource, you will generally want to use it in combination of the -clipdstlayer, -clipdstwhere or -clipdstsql options

-clipdstsql *sql_statement*: Select desired geometries using an SQL query instead.

-clipdstlayer *layername*: Select the named layer from the destination clip datasource.

-clipdstwhere *expression*: Restrict desired geometries based on attribute query.

-wrapdateline: (starting with GDAL 1.7.0) split geometries crossing the dateline meridian (long. = +/- 180deg)

-datelineoffset: (starting with GDAL 1.10) offset from dateline in degrees (default long. = +/- 10deg, geometries within 170deg to -170deg will be split)

-simplify*tolerance*: (starting with GDAL 1.9.0) distance tolerance for simplification. Note: the algorithm used preserves topology per feature, in particular for polygon geometries, but not for a whole layer.

-segmentize*max_dist*: (starting with GDAL 1.6.0) maximum distance between 2 nodes. Used to create intermediate points

-fieldTypeToString*type1, ...*: (starting with GDAL 1.7.0) converts any field of the specified type to a field of type string in the destination layer. Valid types are : Integer, Integer64, Real, String, Date, Time, DateTime, Binary, IntegerList, Integer64List, RealList, StringList. Special value **All** can be used to convert all fields to strings. This is an alternate way to using the CAST operator of OGR SQL, that may avoid typing a long SQL query. Note that this does not influence the field types used by the source driver, and is only an afterwards conversion.

-mapFieldType*srctype*|*All=dsttype, ...*: (starting with GDAL 2.0) converts any field of the specified type to another type. Valid types are : Integer, Integer64, Real, String, Date, Time, DateTime, Binary, IntegerList, Integer64List, RealList, StringList. Types can also include subtype between parenthesis, such as Integer(Boolean), Real(Float32), ... Special value **All** can be used to convert all fields to another type. This is an alternate way to using the CAST operator of OGR SQL, that may avoid typing a long SQL query. This is a generalization of -fieldTypeToString. Note that this does not influence the field types used by the source driver, and is only an afterwards conversion.

-unsetFieldWidth: (starting with GDAL 1.11) set field width and precision to 0.

-splitlistfields: (starting with GDAL 1.8.0) split fields of type StringList, RealList or IntegerList into as many fields of type String, Real or Integer as necessary.

-maxsubfields *val*: To be combined with -splitlistfields to limit the number of subfields created for each split field.

-explodecollections: (starting with GDAL 1.8.0) produce one feature for each geometry in any kind of geometry collection in the source file

-zfield *field_name*: (starting with GDAL 1.8.0) Uses the specified field to fill the Z coordinate of geometries

-gcp *ungeoref_x ungeoref_y georef_x georef_y elevation*: (starting with GDAL 1.10.0) Add the indicated ground control point. This option may be provided multiple times to provide a set of GCPs.

-order *n*: (starting with GDAL 1.10.0) order of polynomial used for warping (1 to 3). The default is to select a polynomial order based on the number of GCPs.

-tps: (starting with GDAL 1.10.0) Force use of thin plate spline transformer based on available GCPs.

-fieldmap: (starting with GDAL 1.10.0) Specifies the list of field indexes to be copied from the source to the destination. The (n)th value specified in the list is the index of the field in the target layer definition in which the n(th) field of the source layer must be copied. Index count starts at zero. There must be exactly as many values in the list as the count of the fields in the source layer. We can use the 'identity' setting to specify that the fields should be transferred by using the same order. This setting should be used along with the -append setting.

-addfields: (starting with GDAL 1.11) This is a specialized version of -append. Contrary to -append, -addfields has the effect of adding, to existing target layers, the new fields found in source layers. This option is useful when merging files that have non-strictly identical structures. This might not work for output formats that don't support adding fields to existing non-empty layers.

-relaxedFieldNameMatch: (starting with GDAL 1.11) Do field name matching between source and existing target layer in a more relaxed way if the target driver has an implementation for it. [-relaxedFieldNameMatch] [-forceNullable]

-forceNullable: (starting with GDAL 2.0) Do not propagate not-nullable constraints to target layer if they exist in source layer..

-unsetDefault: (starting with GDAL 2.0) Do not propagate default field values to target layer if they exist in source layer..

-unsetFid: (starting with GDAL 2.0) Can be specify to prevent the new default behaviour that consists in, if the output driver has a FID layer creation option and we are not in append mode, to preserve the name of the source FID column and source feature IDs

-nomd: (starting with GDAL 2.0) To disable copying of metadata from source dataset and layers into target dataset and layers, when supported by output driver.

-mo *"META-TAG=VALUE"*: (starting with GDAL 2.0) Passes a metadata key and value to set on the output dataset, when supported by output driver.

-noNativeData: (starting with GDAL 2.1) To disable copying of native data, i.e. details of source format not captured by OGR abstraction, that are otherwise preserved by some drivers (like GeoJSON) when converting to same format.

29.3 PERFORMANCE HINTS

When writing into transactional DBMS (SQLite/PostgreSQL,MySQL, etc...), it might be beneficial to increase the number of INSERT statements executed between BEGIN TRANSACTION and COMMIT TRANSACTION statements. This number is specified with the -gt option. For example, for SQLite, explicitly defining **-gt 65536** ensures optimal performance while populating some table containing many hundredth thousand or million rows. However, note that if there are failed insertions, the scope of -skipfailures is a whole transaction.

For PostgreSQL, the PG_USE_COPY config option can be set to YES for a significant insertion performance boost. See the PG driver documentation page.

More generally, consult the documentation page of the input and output drivers for performance hints.

29.4 C API

Starting with GDAL 2.1, this utility is also callable from C with GDALVectorTranslate().

29.5 EXAMPLE

Example appending to an existing layer (both flags need to be used):

```
% ogr2ogr -update -append -f PostgreSQL PG:dbname=warmerda abc.tab
```

Example reprojecting from ETRS_1989_LAEA_52N_10E to EPSG:4326 and clipping to a bounding box

```
% ogr2ogr -wrapdateline -t_srs EPSG:4326 -clipdst -5 40 15 55 france_4326.shp europe_laea.shp
```

Example for using the -fieldmap setting. The first field of the source layer is used to fill the third field (index 2 = third field) of the target layer, the second field of the source layer is ignored, the third field of the source layer used to fill the fifth field of the target layer.

```
% ogr2ogr -append -fieldmap 2,-1,4 dst.shp src.shp
```

More examples are given in the individual format pages.

Chapter 30

ogrtindex

Creates a tileindex.

30.1 SYNOPSIS

```
ogrtindex [-lnum n]... [-lname name]... [-f output_format]
          [-write_absolute_path] [-skip_different_projection]
          [-t_srs target_srs]
          [-src_srs_name field_name] [-src_srs_format [AUTO|WKT|EPSG|PROJ]
          [-accept_different_schemas]
                 output_dataset src_dataset...
```

30.2 DESCRIPTION

The ogrtindex program can be used to create a tileindex - a file containing a list of the identities of a bunch of other files along with there spatial extents. This is primarily intended to be used with MapServer for tiled access to layers using the OGR connection type.

-lnum *n*: Add layer number 'n' from each source file in the tile index.

-lname *name*: Add the layer named 'name' from each source file in the tile index.

-f *output_format*: Select an output format name. The default is to create a shapefile.

-tileindex *field_name*: The name to use for the dataset name. Defaults to LOCATION.

-write_absolute_path: Filenames are written with absolute paths

-skip_different_projection: Only layers with same projection ref as layers already inserted in the tileindex will be inserted.

-t_srs target_srs: (GDAL >= 2.2)

Extent of input files will be transformed to the desired target coordinate reference system. Using this option generates files that are not compatible with MapServer < 7.2. Default creates simple rectangular polygons in the same coordinate reference system as the input rasters.

-src_srs_name field_name: (GDAL >= 2.2)

The name of the field to store the SRS of each tile. This field name can be used as the value of the TILESRS keyword in MapServer >= 7.2.

-src_srs_format type: (GDAL >= 2.2)

The format in which the SRS of each tile must be written. Types can be AUTO, WKT, EPSG, PROJ.

-accept_different_schemas: By default ogrtindex checks that all layers inserted into the index have the same attribute schemas. If you specify this option, this test will be disabled. Be aware that resulting index may be incompatible with MapServer!

If no -lnum or -lname arguments are given it is assumed that all layers in source datasets should be added to the tile index as independent records.

If the tile index already exists it will be appended to, otherwise it will be created.

30.3 EXAMPLE

This example would create a shapefile (tindex.shp) containing a tile index of the BL2000_LINK layers in all the NTF files in the wrk directory:

```
% ogrtindex tindex.shp wrk/*.NTF
```

Chapter 31

ogrlineref

The utility can be used for:

- create linear reference file from input data
- return the "linear referenced" distance for the projection of the input coordinates (point) on the path
- return the coordinates (point) on the path according to the "linear referenced" distance
- return the portion of the path according to the "linear referenced" begin and end distances

31.1 SYNOPSIS

```
ogrlineref [--help-general] [-progress] [-quiet]
           [-f format_name] [[-dsco NAME=VALUE] ...] [[-lco NAME=VALUE]...]
           [-create]
           [-l src_line_datasource_name] [-ln layer_name] [-lf field_name]
           [-p src_repers_datasource_name] [-pn layer_name] [-pm pos_field_name] [-pf field_name]
           [-r src_parts_datasource_name] [-rn layer_name]
           [-o dst_datasource_name] [-on layer_name]  [-of field_name] [-s step]
           [-get_pos] [-x long] [-y lat]
           [-get_coord] [-m position]
           [-get_subline] [-mb position] [-me position]
```

31.2 DESCRIPTION

The ogrlineref program can be used to create a linear reference - a file containing a segments of special length (e.g. 1 km in reference units) and get coordinates, linear referenced distances or sublines (subpaths) from this file. The utility not required the M or Z values in geometry. The results can be stored in any OGR supported format. Also some information is written to the stdout.

--help-general: Show the usage.

-progress: Show progress.

-quiet: Suppress all messages except errors and results.

-f *format_name*: Select an output format name. The default is to create a shapefile.

-dsco *NAME=VALUE*: Dataset creation option (format specific)

-lco*NAME=VALUE*: Layer creation option (format specific)

-create: Create the linear reference file (linestring of parts).

-l*src_line_datasource_name*: The path to input linestring datasource (e.g. the road)

-ln*layer_name*: The layer name in datasource

-lf*field_name*: The field name of unique values to separate the input lines (e.g. the set of roads)

-p*src_repers_datasource_name*: The path to linear references points (e.g. the road mile-stones)

-pn*layer_name*: The layer name in datasource

-pm*pos_field_name*: The field name of distances along path (e.g. mile-stones values)

-pf*field_name*: The field name of unique values to map input reference points to lines

-r*src_parts_datasource_name*: The path to linear reference file

-rn*layer_name*: The layer name in datasource

-o*dst_datasource_name*: The path to output linear reference file (linestring datasource)

-on*layer_name*: The layer name in datasource

-of*field_name*: The field name for storing the unique values of input lines

-s*step*: The part size in linear units

-get_pos: Return linear referenced position for input X, Y

-x*long*: Input X coordinate

-y*lat*: Input Y coordinate

-get_coord: Return point on path for input linear distance

-m*position*: The input linear distance

-get_subline: Return the portion of the input path from and to input linear positions

-mb*position*: The input begin linear distance

-me*position*: The input end linear distance

31.3 EXAMPLE

This example would create a shapefile (parts.shp) containing a data needed for linear referencing (1 km parts):

```
% ogrlineref -create -l roads.shp -p references.shp -pm dist -o parts.shp -s 1000 -progress
```

Chapter 32

Hierarchical Index

32.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 33

Class Index

33.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 34

File Index

34.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 35

Class Documentation

35.1 _curve_data Struct Reference

Public Member Functions

- bool **IsInside** (const double &dfDist) const

Public Attributes

- OGRLineString * **pPart**
- double **dfBeg**
- double **dfEnd**
- double **dfFactor**

The documentation for this struct was generated from the following file:

- ogrlineref.cpp

35.2 AssociatedLayers Struct Reference

Public Attributes

- OGRLayer * **poSrcLayer**
- TargetLayerInfo * **psInfo**

The documentation for this struct was generated from the following file:

- ogr2ogr_lib.cpp

35.3 BandProperty Struct Reference

Public Attributes

- GDALColorInterp **colorInterpretation**
- GDALDataType **dataType**
- GDALColorTableH **colorTable**
- int **bHasNoData**

- double **noDataValue**

The documentation for this struct was generated from the following file:

- gdalbuildvrt_lib.cpp

35.4 ClientInfo Struct Reference

Public Attributes

- int **nSocket**
- void ∗ **hSrvLoopInstance**

The documentation for this struct was generated from the following file:

- gdalserver.c

35.5 ColorAssociation Struct Reference

Public Attributes

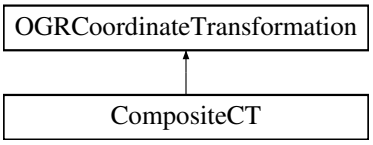
- double **dfVal**
- int **nR**
- int **nG**
- int **nB**
- int **nA**

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.6 CompositeCT Class Reference

Inheritance diagram for CompositeCT:



Public Member Functions

- **CompositeCT** (OGRCoordinateTransformation ∗poCT1In, OGRCoordinateTransformation ∗poCT2In)
- virtual OGRSpatialReference ∗ **GetSourceCS** () override
- virtual OGRSpatialReference ∗ **GetTargetCS** () override
- virtual int **Transform** (int nCount, double ∗x, double ∗y, double ∗z=NULL) override
- virtual int **TransformEx** (int nCount, double ∗x, double ∗y, double ∗z=NULL, int ∗pabSuccess=NULL) override

Public Attributes

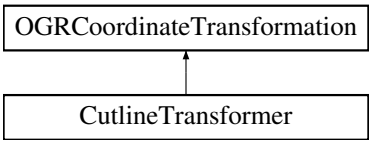
- OGRCoordinateTransformation * **poCT1**
- OGRCoordinateTransformation * **poCT2**

The documentation for this class was generated from the following file:

- ogr2ogr_lib.cpp

35.7 CutlineTransformer Class Reference

Inheritance diagram for CutlineTransformer:



Public Member Functions

- virtual OGRSpatialReference * **GetSourceCS** () override
- virtual OGRSpatialReference * **GetTargetCS** () override
- virtual int **Transform** (int nCount, double *x, double *y, double *z=NULL) override
- virtual int **TransformEx** (int nCount, double *x, double *y, double *z=NULL, int *pabSuccess=NULL) override

Public Attributes

- void * **hSrcImageTransformer**

The documentation for this class was generated from the following file:

- gdalwarp_lib.cpp

35.8 DatasetProperty Struct Reference

Public Attributes

- int **isFileOK**
- int **nRasterXSize**
- int **nRasterYSize**
- double **adfGeoTransform** [6]
- int **nBlockXSize**
- int **nBlockYSize**
- GDALDataType **firstBandType**
- int * **panHasNoData**
- double * **padfNoDataValues**
- int **bHasDatasetMask**
- int **nMaskBlockXSize**
- int **nMaskBlockYSize**

The documentation for this struct was generated from the following file:

- gdalbuildvrt_lib.cpp

35.9 EnhanceCBInfo Struct Reference

Public Attributes

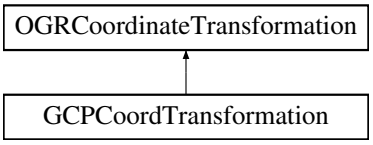
- GDALRasterBand * **poSrcBand**
- GDALDataType **eWrkType**
- double **dfScaleMin**
- double **dfScaleMax**
- int **nLUTBins**
- const int * **panLUT**

The documentation for this struct was generated from the following file:

- gdalenhance.cpp

35.10 GCPCoordTransformation Class Reference

Inheritance diagram for GCPCoordTransformation:



Public Member Functions

- **GCPCoordTransformation** (int nGCPCount, const GDAL_GCP *pasGCPList, int nReqOrder, OGRSpatialReference *poSRSIn)
- bool **IsValid** () const
- virtual OGRSpatialReference * **GetSourceCS** () override
- virtual OGRSpatialReference * **GetTargetCS** () override
- virtual int **Transform** (int nCount, double *x, double *y, double *z=NULL) override
- virtual int **TransformEx** (int nCount, double *x, double *y, double *z=NULL, int *pabSuccess=NULL) override

Public Attributes

- void * **hTransformArg**
- bool **bUseTPS**
- OGRSpatialReference * **poSRS**

The documentation for this class was generated from the following file:

- ogr2ogr_lib.cpp

35.11 GDALAspectAlgData Struct Reference

Public Attributes

- bool **bAngleAsAzimuth**

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.12 GDALBuildVRTOptions Struct Reference

Public Attributes

- char * **pszResolution**
- int **bSeparate**
- int **bAllowProjectionDifference**
- double **we_res**
- double **ns_res**
- int **bTargetAlignedPixels**
- double **xmin**
- double **ymin**
- double **xmax**
- double **ymax**
- int **bAddAlpha**
- int **bHideNoData**
- int **nSubdataset**
- char * **pszSrcNoData**
- char * **pszVRTNoData**
- char * **pszOutputSRS**
- int * **panBandList**
- int **nBandCount**
- int **nMaxBandNo**
- char * **pszResampling**
- char ** **papszOpenOptions**
- int bQuiet
- GDALProgressFunc pfnProgress
- void * pProgressData

35.12.1 Detailed Description

Options for use with GDALBuildVRT(). GDALBuildVRTOptions* must be allocated and freed with GDALBuildVRTOptionsNew() and GDALBuildVRTOptionsFree() respectively.

35.12.2 Member Data Documentation

35.12.2.1 int GDALBuildVRTOptions::bQuiet

allow or suppress progress monitor and other non-error output

35.12.2.2 GDALProgressFunc GDALBuildVRTOptions::pfnProgress

the progress function to use

35.12.2.3 void* GDALBuildVRTOptions::pProgressData

pointer to the progress data variable

The documentation for this struct was generated from the following file:

- gdalbuildvrt_lib.cpp

35.13 GDALBuildVRTOptionsForBinary Struct Reference

Public Attributes

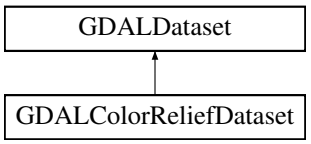
- int **nSrcFiles**
- char ** **papszSrcFiles**
- char * **pszDstFilename**
- int **bQuiet**
- int **bOverwrite**

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.14 GDALColorReliefDataset Class Reference

Inheritance diagram for GDALColorReliefDataset:



Public Member Functions

- **GDALColorReliefDataset** (GDALDatasetH hSrcDS, GDALRasterBandH hSrcBand, const char *pszColorFilename, ColorSelectionMode eColorSelectionMode, int bAlpha)
- bool **InitOK** () const
- CPLErr **GetGeoTransform** (double *padfGeoTransform) override
- const char * **GetProjectionRef** () override

Friends

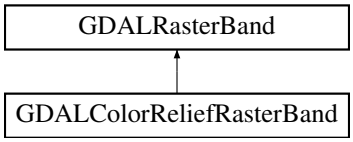
- class **GDALColorReliefRasterBand**

The documentation for this class was generated from the following file:

- gdaldem_lib.cpp

35.15 GDALColorReliefRasterBand Class Reference

Inheritance diagram for GDALColorReliefRasterBand:



Public Member Functions

- **GDALColorReliefRasterBand** (GDALColorReliefDataset *, int)
- virtual CPLErr **IReadBlock** (int, int, void *) override
- virtual GDALColorInterp **GetColorInterpretation** () override

Friends

- class **GDALColorReliefDataset**

The documentation for this class was generated from the following file:

- gdaldem_lib.cpp

35.16 GDALDEMProcessingOptions Struct Reference

Public Attributes

- char * pszFormat
- GDALProgressFunc pfnProgress
- void * pProgressData
- double **z**
- double **scale**
- double **az**
- double **alt**
- int **slopeFormat**
- bool **bAddAlpha**
- bool **bZeroForFlat**
- bool **bAngleAsAzimuth**
- ColorSelectionMode **eColorSelectionMode**
- bool **bComputeAtEdges**
- bool **bZevenbergenThorne**
- bool **bCombined**
- bool **bMultiDirectional**
- char ** **papszCreateOptions**
- int **nBand**

35.16.1 Member Data Documentation

35.16.1.1 GDALProgressFunc GDALDEMProcessingOptions::pfnProgress

the progress function to use

35.16.1.2 void* GDALDEMProcessingOptions::pProgressData

pointer to the progress data variable

35.16.1.3 char* GDALDEMProcessingOptions::pszFormat

output format. The default is GeoTIFF(GTiff). Use the short format name.

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.17 GDALDEMProcessingOptionsForBinary Struct Reference

Public Attributes

- char * **pszProcessing**
- char * **pszSrcFilename**
- char * **pszColorFilename**
- char * **pszDstFilename**
- int **bQuiet**
- int **bFormatExplicitlySet**
- char * **pszFormat**

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.18 GDALError Class Reference

Public Member Functions

- **GDALError** (CPLErr eErr=CE_None, CPLErrorNum errNum=CPLE_None, const char *pszMsg="")

Public Attributes

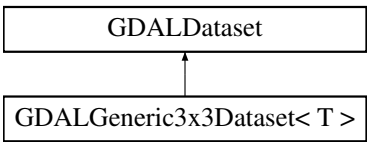
- CPLErr **m_eErr**
- CPLErrorNum **m_errNum**
- CPLString **m_osMsg**

The documentation for this class was generated from the following file:

- gdaladdo.cpp

35.19 GDALGeneric3x3Dataset< T > Class Template Reference

Inheritance diagram for GDALGeneric3x3Dataset< T >:



Public Member Functions

- **GDALGeneric3x3Dataset** (GDALDatasetH hSrcDS, GDALRasterBandH hSrcBand, GDALDataType eDstDataType, int bDstHasNoData, double dfDstNoDataValue, typename GDALGeneric3x3ProcessingAlg< T >::type pfnAlg, void *pAlgData, bool bComputeAtEdges)
- bool **InitOK** () const
- CPLErr **GetGeoTransform** (double *padfGeoTransform) override
- const char * **GetProjectionRef** () override

Friends

- class **GDALGeneric3x3RasterBand**< **T** >

The documentation for this class was generated from the following file:

- gdaldem_lib.cpp

35.20 GDALGeneric3x3ProcessingAlg< T > Struct Template Reference

Public Types

- typedef float(* **type**)(const T *pafWindow, float fDstNoDataValue, void *pData)

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.21 GDALGeneric3x3ProcessingAlg_multisample< T > Struct Template Reference

Public Types

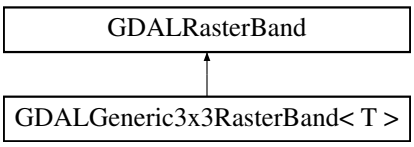
- typedef int(* **type**)(const T *pafThreeLineWin, int nLine1Off, int nLine2Off, int nLine3Off, int nXSize, void *pData, float *pafOutputBuf)

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.22 GDALGeneric3x3RasterBand< T > Class Template Reference

Inheritance diagram for GDALGeneric3x3RasterBand< T >:



Public Member Functions

- **GDALGeneric3x3RasterBand** (GDALGeneric3x3Dataset< T > *poDS, GDALDataType eDstDataType)
- virtual CPLErr **IReadBlock** (int, int, void *) override
- virtual double **GetNoDataValue** (int *pbHasNoData) override

Friends

- class **GDALGeneric3x3Dataset< T >**

The documentation for this class was generated from the following file:

- gdaldem_lib.cpp

35.23 GDALGridOptions Struct Reference

Public Attributes

- char * pszFormat
- int bQuiet
- GDALProgressFunc pfnProgress
- void * pProgressData
- char ** **papszLayers**
- char * **pszBurnAttribute**
- double **dfIncreaseBurnValue**
- double **dfMultiplyBurnValue**
- char * **pszWHERE**
- char * **pszSQL**
- GDALDataType **eOutputType**
- char ** **papszCreateOptions**
- int **nXSize**
- int **nYSize**
- double **dfXMin**
- double **dfXMax**
- double **dfYMin**
- double **dfYMax**
- int **bIsXExtentSet**
- int **bIsYExtentSet**
- GDALGridAlgorithm **eAlgorithm**
- void * **pOptions**
- char * **pszOutputSRS**
- OGRGeometry * **poSpatialFilter**
- int **bClipSrc**
- OGRGeometry * **poClipSrc**
- char * **pszClipSrcDS**
- char * **pszClipSrcSQL**

- char * **pszClipSrcLayer**
- char * **pszClipSrcWhere**
- int **bNoDataSet**
- double **dfNoDataValue**

35.23.1 Detailed Description

Options for use with GDALGrid(). GDALGridOptions* must be allocated and freed with GDALGridOptionsNew() and GDALGridOptionsFree() respectively.

35.23.2 Member Data Documentation

35.23.2.1 int GDALGridOptions::bQuiet

allow or suppress progress monitor and other non-error output

35.23.2.2 GDALProgressFunc GDALGridOptions::pfnProgress

the progress function to use

35.23.2.3 void* GDALGridOptions::pProgressData

pointer to the progress data variable

35.23.2.4 char* GDALGridOptions::pszFormat

output format. The default is GeoTIFF(GTiff). Use the short format name.

The documentation for this struct was generated from the following file:

- gdal_grid_lib.cpp

35.24 GDALGridOptionsForBinary Struct Reference

Public Attributes

- char * **pszSource**
- char * **pszDest**
- int **bQuiet**
- int **bFormatExplicitlySet**
- char * **pszFormat**

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.25 GDALHillshadeAlgData Struct Reference

Public Attributes

- double **inv_nsres**
- double **inv_ewres**
- double **sin_altRadians**
- double **cos_alt_mul_z**
- double **azRadians**
- double **cos_az_mul_cos_alt_mul_z**
- double **sin_az_mul_cos_alt_mul_z**
- double **square_z**
- double **sin_altRadians_mul_254**
- double **cos_az_mul_cos_alt_mul_z_mul_254**
- double **sin_az_mul_cos_alt_mul_z_mul_254**
- double **square_z_mul_square_inv_res**
- double **cos_az_mul_cos_alt_mul_z_mul_254_mul_inv_res**
- double **sin_az_mul_cos_alt_mul_z_mul_254_mul_inv_res**

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.26 GDALHillshadeMultiDirectionalAlgData Struct Reference

Public Attributes

- double **inv_nsres**
- double **inv_ewres**
- double **square_z**
- double **sin_altRadians_mul_127**
- double **sin_altRadians_mul_254**
- double **cos_alt_mul_z_mul_127**
- double **cos225_az_mul_cos_alt_mul_z_mul_127**

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.27 GDALInfoOptions Struct Reference

Public Attributes

- GDALInfoFormat eFormat
- int **bComputeMinMax**
- int bReportHistograms
- int bReportProj4
- int bStats
- int bApproxStats
- int **bSample**
- int bComputeChecksum
- int bShowGCPs

- int bShowMetadata
- int bShowRAT
- int bShowColorTable
- int bListMDD
- int bShowFileList
- char ** papszExtraMDDomains
- bool **bStdoutOutput**

35.27.1 Detailed Description

Options for use with GDALInfo(). GDALInfoOptions* must be allocated and freed with GDALInfoOptionsNew() and GDALInfoOptionsFree() respectively.

35.27.2 Member Data Documentation

35.27.2.1 int GDALInfoOptions::bApproxStats

read and display image statistics. Force computation if no statistics are stored in an image. However, they may be computed based on overviews or a subset of all tiles. Useful if you are in a hurry and don't want precise stats.

35.27.2.2 int GDALInfoOptions::bComputeChecksum

force computation of the checksum for each band in the dataset

35.27.2.3 int GDALInfoOptions::bListMDD

list all metadata domains available for the dataset

35.27.2.4 int GDALInfoOptions::bReportHistograms

report histogram information for all bands

35.27.2.5 int GDALInfoOptions::bReportProj4

report a PROJ.4 string corresponding to the file's coordinate system

35.27.2.6 int GDALInfoOptions::bShowColorTable

allow or suppress printing of color table

35.27.2.7 int GDALInfoOptions::bShowFileList

display the file list or the first file of the file list

35.27.2.8 int GDALInfoOptions::bShowGCPs

allow or suppress ground control points list printing. It may be useful for datasets with huge amount of GCPs, such as L1B AVHRR or HDF4 MODIS which contain thousands of them.

35.27.2.9 int GDALInfoOptions::bShowMetadata

allow or suppress metadata printing. Some datasets may contain a lot of metadata strings.

35.27.2.10 int GDALInfoOptions::bShowRAT

allow or suppress printing of raster attribute table

35.27.2.11 int GDALInfoOptions::bStats

read and display image statistics. Force computation if no statistics are stored in an image

35.27.2.12 GDALInfoFormat GDALInfoOptions::eFormat

output format

35.27.2.13 char** GDALInfoOptions::papszExtraMDDomains

report metadata for the specified domains. "all" can be used to report metadata in all domains.

The documentation for this struct was generated from the following file:

- gdalinfo_lib.cpp

35.28 GDALInfoOptionsForBinary Struct Reference

Public Attributes

- char * **pszFilename**
- char ** **papszOpenOptions**
- int **nSubdataset**

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.29 GDALNearblackOptions Struct Reference

Public Attributes

- char * pszFormat
- GDALProgressFunc pfnProgress
- void * pProgressData
- int **nMaxNonBlack**
- int **nNearDist**
- int **bNearWhite**
- int **bSetAlpha**
- int **bSetMask**
- Colors **oColors**
- char ** **papszCreationOptions**

35.29.1 Member Data Documentation

35.29.1.1 GDALProgressFunc GDALNearblackOptions::pfnProgress

the progress function to use

35.29.1.2 void* GDALNearblackOptions::pProgressData

pointer to the progress data variable

35.29.1.3 char* GDALNearblackOptions::pszFormat

output format. The default is GeoTIFF(GTiff). Use the short format name.

The documentation for this struct was generated from the following file:

- nearblack_lib.cpp

35.30 GDALNearblackOptionsForBinary Struct Reference

Public Attributes

- char * **pszInFile**
- char * **pszOutFile**
- int **bQuiet**
- int **bFormatExplicitlySet**
- char * **pszFormat**

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.31 GDALRasterizeOptions Struct Reference

Public Attributes

- char * pszFormat
- GDALProgressFunc pfnProgress
- void * pProgressData
- int **bCreateOutput**
- int **b3D**
- int **bInverse**
- char ** **papszLayers**
- char * **pszSQL**
- char * **pszDialect**
- char * **pszBurnAttribute**
- char * **pszWHERE**
- std::vector< int > **anBandList**
- std::vector< double > **adfBurnValues**
- char ** **papszRasterizeOptions**
- double **dfXRes**
- double **dfYRes**

- char ** **papszCreationOptions**
- GDALDataType **eOutputType**
- std::vector< double > **adfInitVals**
- int **bNoDataSet**
- double **dfNoData**
- OGREnvelope **sEnvelop**
- int **bGotBounds**
- int **nXSize**
- int **nYSize**
- OGRSpatialReferenceH **hSRS**
- int **bTargetAlignedPixels**

35.31.1 Member Data Documentation

35.31.1.1 GDALProgressFunc GDALRasterizeOptions::pfnProgress

the progress function to use

35.31.1.2 void* GDALRasterizeOptions::pProgressData

pointer to the progress data variable

35.31.1.3 char* GDALRasterizeOptions::pszFormat

output format. The default is GeoTIFF(GTiff). Use the short format name.

The documentation for this struct was generated from the following file:

- gdal_rasterize_lib.cpp

35.32 GDALRasterizeOptionsForBinary Struct Reference

Public Attributes

- char * **pszSource**
- char * **pszDest**
- int **bQuiet**
- int **bFormatExplicitlySet**
- char * **pszFormat**
- int **bCreateOutput**

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.33 GDALSlopeAlgData Struct Reference

Public Attributes

- double **nsres**
- double **ewres**

- double **scale**
- int **slopeFormat**

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.34 GDALTranslateOptions Struct Reference

Public Attributes

- char * pszFormat
- int bQuiet
- GDALProgressFunc pfnProgress
- void * pProgressData
- GDALDataType eOutputType
- MaskMode **eMaskMode**
- int nBandCount
- int * panBandList
- int nOXSizePixel
- int **nOYSizePixel**
- double dfOXSizePct
- double **dfOYSizePct**
- char ** papszCreateOptions
- double adfSrcWin [4]
- int bStrict
- int bUnscale
- int nScaleRepeat
- GDALTranslateScaleParams * pasScaleParams
- int bHasUsedExplicitScaleBand
- int nExponentRepeat
- double * padfExponent
- int **bHasUsedExplicitExponentBand**
- char ** papszMetadataOptions
- char * pszOutputSRS
- int nGCPCount
- GDAL_GCP * pasGCPs
- double adfULLR [4]
- int bSetNoData
- int bUnsetNoData
- double dfNoDataReal
- int nRGBExpand
- int **nMaskBand**
- int bStats
- int **bApproxStats**
- int bErrorOnPartiallyOutside
- int bErrorOnCompletelyOutside
- int bNoRAT
- char * pszResampling
- double dfXRes
- double **dfYRes**
- double dfULX
- double **dfULY**
- double **dfLRX**
- double **dfLRY**
- char * pszProjSRS

35.34.1 Detailed Description

Options for use with GDALTranslate(). GDALTranslateOptions* must be allocated and freed with GDALTranslateOptionsNew() and GDALTranslateOptionsFree() respectively.

35.34.2 Member Data Documentation

35.34.2.1 double GDALTranslateOptions::adfSrcWin[4]

subwindow from the source image for copying based on pixel/line location

35.34.2.2 double GDALTranslateOptions::adfULLR[4]

assign/override the georeferenced bounds of the output file. This assigns georeferenced bounds to the output file, ignoring what would have been derived from the source file. So this does not cause reprojection to the specified SRS.

35.34.2.3 int GDALTranslateOptions::bErrorOnCompletelyOutside

Same as bErrorOnPartiallyOutside, except that the criterion for erroring out is when the request falls completely outside the source raster extent.

35.34.2.4 int GDALTranslateOptions::bErrorOnPartiallyOutside

If this option is set, GDALTranslateOptions::adfSrcWin or (GDALTranslateOptions::dfULX, GDALTranslateOptions::dfULY, GDALTranslateOptions::dfLRX, GDALTranslateOptions::dfLRY) values that falls partially outside the source raster extent will be considered as an error. The default behaviour is to accept such requests.

35.34.2.5 int GDALTranslateOptions::bHasUsedExplicitScaleBand

It is set to TRUE, when scale parameters are specific to each band

35.34.2.6 int GDALTranslateOptions::bNoRAT

does not copy source RAT into destination dataset (when TRUE)

35.34.2.7 int GDALTranslateOptions::bQuiet

allow or suppress progress monitor and other non-error output

35.34.2.8 int GDALTranslateOptions::bSetNoData

set a nodata value specified in GDALTranslateOptions::dfNoDataReal to the output bands

35.34.2.9 int GDALTranslateOptions::bStats

force recomputation of statistics

35.34.2.10 int GDALTranslateOptions::bStrict

don't be forgiving of mismatches and lost data when translating to the output format

35.34.2.11 int GDALTranslateOptions::bUnscale

apply the scale/offset metadata for the bands to convert scaled values to unscaled values. It is also often necessary to reset the output datatype with GDALTranslateOptions::eOutputType

35.34.2.12 int GDALTranslateOptions::bUnsetNoData

avoid setting a nodata value to the output file if one exists for the source file

35.34.2.13 double GDALTranslateOptions::dfNoDataReal

Assign a specified nodata value to output bands (GDALTranslateOptions::bSetNoData option should be set). Note that if the input dataset has a nodata value, this does not cause pixel values that are equal to that nodata value to be changed to the value specified.

35.34.2.14 double GDALTranslateOptions::dfOXSizePct

size of the output file. GDALTranslateOptions::dfOXSizePct and GDALTranslateOptions::dfOYSizePct are fraction of the input image size. The value 100 means 100%. If one of the two values is set to 0, its value will be determined from the other one, while maintaining the aspect ratio of the source dataset

35.34.2.15 double GDALTranslateOptions::dfULX

subwindow from the source image for copying (like GDALTranslateOptions::adfSrcWin) but with the corners given in georeferenced coordinates (by default expressed in the SRS of the dataset. Can be changed with pszProjSRS)

35.34.2.16 double GDALTranslateOptions::dfXRes

target resolution. The values must be expressed in georeferenced units. Both must be positive values. This is exclusive with GDALTranslateOptions::nOXSizePixel (or GDALTranslateOptions::dfOXSizePct), GDALTranslateOptions::nOYSizePixel (or GDALTranslateOptions::dfOYSizePct) and GDALTranslateOptions::adfULLR

35.34.2.17 GDALDataType GDALTranslateOptions::eOutputType

for the output bands to be of the indicated data type

35.34.2.18 int GDALTranslateOptions::nBandCount

number of input bands to write to the output file, or to reorder bands

35.34.2.19 int GDALTranslateOptions::nExponentRepeat

the size of the list padfExponent

35.34.2.20 int GDALTranslateOptions::nGCPCount

number of GCPS to be added to the output dataset

35.34.2.21 int GDALTranslateOptions::nOXSizePixel

size of the output file. GDALTranslateOptions::nOXSizePixel is in pixels and GDALTranslateOptions::nOYSizePixel is in lines. If one of the two values is set to 0, its value will be determined from the other one, while maintaining the aspect ratio of the source dataset

35.34.2.22 int GDALTranslateOptions::nRGBExpand

to expose a dataset with 1 band with a color table as a dataset with 3 (RGB) or 4 (RGBA) bands. Useful for output drivers such as JPEG, JPEG2000, MrSID, ECW that don't support color indexed datasets. The 1 value enables to expand a dataset with a color table that only contains gray levels to a gray indexed dataset.

35.34.2.23 int GDALTranslateOptions::nScaleRepeat

the size of pasScaleParams

35.34.2.24 double* GDALTranslateOptions::padfExponent

to apply non-linear scaling with a power function. It is the list of exponents of the power function (must be positive). This option must be used with GDALTranslateOptions::pasScaleParams. If GDALTranslateOptions::nExponentRepeat is 1, it is applied to all bands of the output image.

35.34.2.25 int* GDALTranslateOptions::panBandList

list of input bands to write to the output file, or to reorder bands. The value 1 corresponds to the 1st band.

35.34.2.26 char** GDALTranslateOptions::papszCreateOptions

list of creation options to the output format driver

35.34.2.27 char** GDALTranslateOptions::papszMetadataOptions

list of metadata key and value to set on the output dataset if possible. GDALTranslateOptionsSetMetadataOptions() and GDALTranslateOptionsAddMetadataOptions() should be used

35.34.2.28 GDAL_GCP* GDALTranslateOptions::pasGCPs

list of GCPs to be added to the output dataset

35.34.2.29 GDALTranslateScaleParams* GDALTranslateOptions::pasScaleParams

the list of scale parameters for each band.

35.34.2.30 GDALProgressFunc GDALTranslateOptions::pfnProgress

the progress function to use

35.34.2.31 void* GDALTranslateOptions::pProgressData

pointer to the progress data variable

35.34.2.32 char* GDALTranslateOptions::pszFormat

output format. The default is GeoTIFF(GTiff). Use the short format name.

35.34.2.33 char* GDALTranslateOptions::pszOutputSRS

override the projection for the output file. The SRS may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.

35.34.2.34 char* GDALTranslateOptions::pszProjSRS

SRS in which to interpret the coordinates given with GDALTranslateOptions::dfULX, GDALTranslateOptions::dfULY, GDALTranslateOptions::dfLRX, GDALTranslateOptions::dfLRY. The SRS may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT. Note that this does not cause reprojection of the dataset to the specified SRS.

35.34.2.35 char* GDALTranslateOptions::pszResampling

resampling algorithm nearest (default), bilinear, cubic, cubicspline, lanczos, average, mode

The documentation for this struct was generated from the following file:

- gdal_translate_lib.cpp

35.35 GDALTranslateOptionsForBinary Struct Reference

Public Attributes

- char * **pszSource**
- char * **pszDest**
- int **bQuiet**
- int **bCopySubDatasets**
- char ** **papszOpenOptions**
- int **bFormatExplicitlySet**
- char * **pszFormat**

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.36 GDALTranslateScaleParams Struct Reference

Public Attributes

- int bScale
- int bHaveScaleSrc
- double dfScaleSrcMin
- double **dfScaleSrcMax**
- double dfScaleDstMin
- double **dfScaleDstMax**

35.36.1 Detailed Description

scaling parameters for use in GDALTranslateOptions.

35.36.2 Member Data Documentation

35.36.2.1 int GDALTranslateScaleParams::bHaveScaleSrc

set it to TRUE if dfScaleSrcMin and dfScaleSrcMax is set. When it is FALSE, the input range is automatically computed from the source data.

35.36.2.2 int GDALTranslateScaleParams::bScale

scaling is done only if it is set to TRUE. This is helpful when there is a need to scale only certain bands.

35.36.2.3 double GDALTranslateScaleParams::dfScaleDstMin

the range of output pixel values. If GDALTranslateScaleParams::dfScaleDstMin and GDALTranslateScaleParams-::dfScaleDstMax are not set, then the output range is 0 to 255.

35.36.2.4 double GDALTranslateScaleParams::dfScaleSrcMin

the range of input pixel values which need to be scaled

The documentation for this struct was generated from the following file:

- gdal_translate_lib.cpp

35.37 GDALVectorTranslateOptions Struct Reference

Public Attributes

- bool bSkipFailures
- int nLayerTransaction
- bool bForceTransaction
- int nGroupTransactions
- GIntBig nFIDToFetch
- bool bQuiet
- char * pszFormat
- char ** papszLayers
- char ** papszDSCO
- char ** papszLCO
- GDALVectorTranslateAccessMode eAccessMode
- bool bAddMissingFields
- bool bTransform
- char * pszOutputSRSDef
- char * pszSourceSRSDef
- bool **bNullifyOutputSRS**
- bool bExactFieldNameMatch
- char * pszNewLayerName
- char * pszWHERE
- char * pszGeomField

- char ** papszSelFields
- char * pszSQLStatement
- char * pszDialect
- int eGType
- GeomTypeConversion **eGeomTypeConversion**
- GeomOperation eGeomOp
- double dfGeomOpParam
- char ** papszFieldTypesToString
- char ** papszMapFieldType
- bool bUnsetFieldWidth
- bool bDisplayProgress
- bool bWrapDateline
- double dfDateLineOffset
- bool bClipSrc
- OGRGeometryH **hClipSrc**
- char * pszClipSrcDS
- char * pszClipSrcSQL
- char * pszClipSrcLayer
- char * pszClipSrcWhere
- OGRGeometryH **hClipDst**
- char * pszClipDstDS
- char * pszClipDstSQL
- char * pszClipDstLayer
- char * pszClipDstWhere
- bool bSplitListFields
- int nMaxSplitListSubFields
- bool bExplodeCollections
- char * pszZField
- char ** papszFieldMap
- int nCoordDim
- char ** papszDestOpenOptions
- bool bForceNullable
- bool bUnsetDefault
- bool bUnsetFid
- bool bPreserveFID
- bool bCopyMD
- char ** papszMetadataOptions
- char * pszSpatSRSDef
- int nGCPCount
- GDAL_GCP * pasGCPs
- int nTransformOrder
- OGRGeometryH hSpatialFilter
- GDALProgressFunc pfnProgress
- void * pProgressData
- bool bNativeData
- GIntBig nLimit

35.37.1 Detailed Description

Options for use with GDALVectorTranslate(). GDALVectorTranslateOptions* must be allocated and freed with GDALVectorTranslateOptionsNew() and GDALVectorTranslateOptionsFree() respectively.

35.37.2 Member Data Documentation

35.37.2.1 bool GDALVectorTranslateOptions::bAddMissingFields

It has the effect of adding, to existing target layers, the new fields found in source layers. This option is useful when merging files that have non-strictly identical structures. This might not work for output formats that don't support adding fields to existing non-empty layers.

35.37.2.2 bool GDALVectorTranslateOptions::bClipSrc

clip geometries when it is set to true

35.37.2.3 bool GDALVectorTranslateOptions::bCopyMD

set it to false to disable copying of metadata from source dataset and layers into target dataset and layers, when supported by output driver.

35.37.2.4 bool GDALVectorTranslateOptions::bDisplayProgress

```
display progress on terminal. Only works if input layers have the "fast feature count"
```

capability

35.37.2.5 bool GDALVectorTranslateOptions::bExactFieldNameMatch

If set to false, then field name matching between source and existing target layer is done in a more relaxed way if the target driver has an implementation for it.

35.37.2.6 bool GDALVectorTranslateOptions::bExplodeCollections

produce one feature for each geometry in any kind of geometry collection in the source file

35.37.2.7 bool GDALVectorTranslateOptions::bForceNullable

If set to true, does not propagate not-nullable constraints to target layer if they exist in source layer

35.37.2.8 bool GDALVectorTranslateOptions::bForceTransaction

force the use of particular transaction type based on GDALVectorTranslate::nLayerTransaction

35.37.2.9 bool GDALVectorTranslateOptions::bNativeData

Whether layer and feature native data must be transferred.

35.37.2.10 bool GDALVectorTranslateOptions::bPreserveFID

use the FID of the source features instead of letting the output driver to automatically assign a new one. If not in append mode, this behaviour becomes the default if the output driver has a FID layer creation option. In which case the name of the source FID column will be used and source feature IDs will be attempted to be preserved. This behaviour can be disabled by option GDALVectorTranslateOptions::bUnsetFid

35.37.2.11 bool GDALVectorTranslateOptions::bQuiet

allow or suppress progress monitor and other non-error output

35.37.2.12 bool GDALVectorTranslateOptions::bSkipFailures

continue after a failure, skipping the failed feature

35.37.2.13 bool GDALVectorTranslateOptions::bSplitListFields

split fields of type StringList, RealList or IntegerList into as many fields of type String, Real or Integer as necessary.

35.37.2.14 bool GDALVectorTranslateOptions::bTransform

It must be set to true to trigger reprojection, otherwise only SRS assignment is done.

35.37.2.15 bool GDALVectorTranslateOptions::bUnsetDefault

If set to true, does not propagate default field values to target layer if they exist in source layer

35.37.2.16 bool GDALVectorTranslateOptions::bUnsetFid

to prevent the new default behaviour that consists in, if the output driver has a FID layer creation option and we are not in append mode, to preserve the name of the source FID column and source feature IDs

35.37.2.17 bool GDALVectorTranslateOptions::bUnsetFieldWidth

set field width and precision to 0

35.37.2.18 bool GDALVectorTranslateOptions::bWrapDateline

split geometries crossing the dateline meridian

35.37.2.19 double GDALVectorTranslateOptions::dfDateLineOffset

```
offset from dateline in degrees (default long. = +/- 10deg, geometries
```

within 170deg to -170deg will be split)

35.37.2.20 double GDALVectorTranslateOptions::dfGeomOpParam

the parameter to geometric operation

35.37.2.21 GDALVectorTranslateAccessMode GDALVectorTranslateOptions::eAccessMode

access modes

35.37.2.22 GeomOperation GDALVectorTranslateOptions::eGeomOp

Geometric operation to perform

35.37.2.23 int GDALVectorTranslateOptions::eGType

the geometry type for the created layer

35.37.2.24 OGRGeometryH GDALVectorTranslateOptions::hSpatialFilter

spatial query extents, in the SRS of the source layer(s) (or the one specified with GDALVectorTranslateOptions::pszSpatSRSDef). Only features whose geometry intersects the extents will be selected. The geometries will not be clipped unless GDALVectorTranslateOptions::bClipSrc is true.

35.37.2.25 int GDALVectorTranslateOptions::nCoordDim

force the coordinate dimension to nCoordDim (valid values are 2 or 3). This affects both the layer geometry type, and feature geometries.

35.37.2.26 GIntBig GDALVectorTranslateOptions::nFIDToFetch

If provided, only the feature with this feature id will be reported. Operates exclusive of the spatial or attribute queries. Note: if you want to select several features based on their feature id, you can also use the fact the 'fid' is a special field recognized by OGR SQL. So GDALVectorTranslateOptions::pszWHERE = "fid in (1,3,5)" would select features 1, 3 and 5.

35.37.2.27 int GDALVectorTranslateOptions::nGCPCount

size of the list GDALVectorTranslateOptions::pasGCPs

35.37.2.28 int GDALVectorTranslateOptions::nGroupTransactions

group nGroupTransactions features per transaction (default 20000). Increase the value for better performance when writing into DBMS drivers that have transaction support. nGroupTransactions can be set to -1 to load the data into a single transaction

35.37.2.29 int GDALVectorTranslateOptions::nLayerTransaction

use layer level transaction. If set to FALSE, then it is interpreted as dataset level transaction.

35.37.2.30 GIntBig GDALVectorTranslateOptions::nLimit

Maximum number of features, or -1 if no limit.

35.37.2.31 int GDALVectorTranslateOptions::nMaxSplitListSubFields

limit the number of subfields created for each split field.

35.37.2.32 int GDALVectorTranslateOptions::nTransformOrder

order of polynomial used for warping (1 to 3). The default is to select a polynomial order based on the number of GCPs

35.37.2.33 char** GDALVectorTranslateOptions::papszDestOpenOptions

destination dataset open option (format specific), only valid in update mode

35.37.2.34 char** GDALVectorTranslateOptions::papszDSCO

dataset creation option (format specific)

35.37.2.35 char** GDALVectorTranslateOptions::papszFieldMap

the list of field indexes to be copied from the source to the destination. The (n)th value specified in the list is the index of the field in the target layer definition in which the n(th) field of the source layer must be copied. Index count starts at zero. There must be exactly as many values in the list as the count of the fields in the source layer. We can use the "identity" option to specify that the fields should be transferred by using the same order. This option should be used along with the GDALVectorTranslateOptions::eAccessMode = ACCESS_APPEND option.

35.37.2.36 char** GDALVectorTranslateOptions::papszFieldTypesToString

list of field types to convert to a field of type string in the destination layer. Valid types are: Integer, Integer64, Real, String, Date, Time, DateTime, Binary, IntegerList, Integer64List, RealList, StringList. Special value "All" can be used to convert all fields to strings. This is an alternate way to using the CAST operator of OGR SQL, that may avoid typing a long SQL query. Note that this does not influence the field types used by the source driver, and is only an afterwards conversion.

35.37.2.37 char** GDALVectorTranslateOptions::papszLayers

list of layers of the source dataset which needs to be selected

35.37.2.38 char** GDALVectorTranslateOptions::papszLCO

layer creation option (format specific)

35.37.2.39 char** GDALVectorTranslateOptions::papszMapFieldType

list of field types and the field type after conversion in the destination layer. ("srctype1=dsttype1","srctype2=dsttype2",...). Valid types are : Integer, Integer64, Real, String, Date, Time, DateTime, Binary, IntegerList, Integer64List, RealList, StringList. Types can also include subtype between parenthesis, such as Integer(Boolean), Real(Float32), ... Special value "All" can be used to convert all fields to another type. This is an alternate way to using the CAST operator of OGR SQL, that may avoid typing a long SQL query. This is a generalization of GDALVectorTranslate-Options::papszFieldTypeToString. Note that this does not influence the field types used by the source driver, and is only an afterwards conversion.

35.37.2.40 char** GDALVectorTranslateOptions::papszMetadataOptions

list of metadata key and value to set on the output dataset, when supported by output driver. ("META-TAG1=VAL-UE1","META-TAG2=VALUE2")

35.37.2.41 char** GDALVectorTranslateOptions::papszSelFields

list of fields from input layer to copy to the new layer. A field is skipped if mentioned previously in the list even if the input layer has duplicate field names. (Defaults to all; any field is skipped if a subsequent field with same name is found.) Geometry fields can also be specified in the list.

35.37.2.42 GDAL_GCP* GDALVectorTranslateOptions::pasGCPs

list of ground control points to be added

35.37.2.43 GDALProgressFunc GDALVectorTranslateOptions::pfnProgress

the progress function to use

35.37.2.44 void* GDALVectorTranslateOptions::pProgressData

pointer to the progress data variable

35.37.2.45 char* GDALVectorTranslateOptions::pszClipDstDS

destination clip datasource

35.37.2.46 char* GDALVectorTranslateOptions::pszClipDstLayer

selected named layer from the destination clip datasource

35.37.2.47 char* GDALVectorTranslateOptions::pszClipDstSQL

select desired geometries using an SQL query

35.37.2.48 char* GDALVectorTranslateOptions::pszClipDstWhere

restrict desired geometries based on attribute query

35.37.2.49 char* GDALVectorTranslateOptions::pszClipSrcDS

clip datasource

35.37.2.50 char* GDALVectorTranslateOptions::pszClipSrcLayer

selected named layer from the source clip datasource

35.37.2.51 char* GDALVectorTranslateOptions::pszClipSrcSQL

select desired geometries using an SQL query

35.37.2.52 char* GDALVectorTranslateOptions::pszClipSrcWhere

restrict desired geometries based on attribute query

35.37.2.53 char* GDALVectorTranslateOptions::pszDialect

SQL dialect. In some cases can be used to use (unoptimized) OGR SQL instead of the native SQL of an RDBMS by using "OGRSQL". The "SQLITE" dialect can also be used with any datasource.

35.37.2.54 char* GDALVectorTranslateOptions::pszFormat

output file format name (default is ESRI Shapefile)

35.37.2.55 char* GDALVectorTranslateOptions::pszGeomField

name of the geometry field on which the spatial filter operates on.

35.37.2.56 char* GDALVectorTranslateOptions::pszNewLayerName

an alternate name to the new layer

35.37.2.57 char* GDALVectorTranslateOptions::pszOutputSRSDef

output SRS. GDALVectorTranslateOptions::bTransform must be set to true to trigger reprojection, otherwise only SRS assignment is done.

35.37.2.58 char* GDALVectorTranslateOptions::pszSourceSRSDef

override source SRS

35.37.2.59 char* GDALVectorTranslateOptions::pszSpatSRSDef

override spatial filter SRS

35.37.2.60 char* GDALVectorTranslateOptions::pszSQLStatement

SQL statement to execute. The resulting table/layer will be saved to the output.

35.37.2.61 char* GDALVectorTranslateOptions::pszWHERE

attribute query (like SQL WHERE)

35.37.2.62 char* GDALVectorTranslateOptions::pszZField

uses the specified field to fill the Z coordinates of geometries

The documentation for this struct was generated from the following file:

- ogr2ogr_lib.cpp

35.38 GDALVectorTranslateOptionsForBinary Struct Reference

Public Attributes

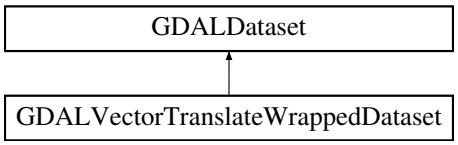
- char * **pszDataSource**
- char * **pszDestDataSource**
- int **bQuiet**
- char ** **papszOpenOptions**
- int **bFormatExplicitlySet**
- char * **pszFormat**
- GDALVectorTranslateAccessMode **eAccessMode**

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.39 GDALVectorTranslateWrappedDataset Class Reference

Inheritance diagram for GDALVectorTranslateWrappedDataset:



Public Member Functions

- virtual int **GetLayerCount** () override
- virtual OGRLayer * **GetLayer** (int nIdx) override
- virtual OGRLayer * **GetLayerByName** (const char *pszName) override
- virtual OGRLayer * **ExecuteSQL** (const char *pszStatement, OGRGeometry *poSpatialFilter, const char *pszDialect) override
- virtual void **ReleaseResultSet** (OGRLayer *poResultsSet) override

Static Public Member Functions

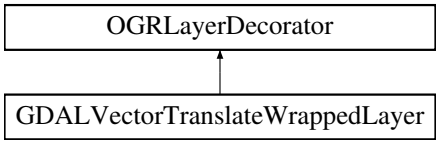
- static GDALVectorTranslateWrappedDataset * **New** (GDALDataset *poBase, OGRSpatialReference *poOutputSRS, bool bTransform)

The documentation for this class was generated from the following file:

- ogr2ogr_lib.cpp

35.40 GDALVectorTranslateWrappedLayer Class Reference

Inheritance diagram for GDALVectorTranslateWrappedLayer:



Public Member Functions

- virtual OGRFeatureDefn * **GetLayerDefn** () override
- virtual OGRFeature * **GetNextFeature** () override
- virtual OGRFeature * **GetFeature** (GIntBig nFID) override

Static Public Member Functions

- static GDALVectorTranslateWrappedLayer * **New** (OGRLayer *poBaseLayer, bool bOwnBaseLayer, OGRSpatialReference *poOutputSRS, bool bTransform)

The documentation for this class was generated from the following file:

- ogr2ogr_lib.cpp

35.41 GDALWarpAppOptions Struct Reference

Public Attributes

- double dfMinX
- double **dfMinY**
- double **dfMaxX**
- double **dfMaxY**
- char * pszTE_SRS
- double dfXRes
- double **dfYRes**
- int bTargetAlignedPixels
- int nForcePixels
- int **nForceLines**
- int bQuiet
- GDALProgressFunc pfnProgress
- void * pProgressData
- bool bEnableDstAlpha
- bool bEnableSrcAlpha
- bool bDisableSrcAlpha
- char * pszFormat
- int **bCreateOutput**
- char ** papszWarpOptions
- double **dfErrorThreshold**
- double dfWarpMemoryLimit
- char ** papszCreateOptions
- GDALDataType eOutputType
- GDALDataType eWorkingType
- GDALResampleAlg eResampleAlg
- char * pszSrcNodata
- char * pszDstNodata
- int bMulti
- char ** papszTO
- char * pszCutlineDSName
- char * pszCLayer
- char * pszCWHERE
- char * pszCSQL

- int bCropToCutline
- int bCopyMetadata
- int bCopyBandInfo
- char * pszMDConflictValue
- bool bSetColorInterpretation
- int nOvLevel
- bool bNoVShiftGrid

35.41.1 Detailed Description

Options for use with GDALWarp(). GDALWarpAppOptions* must be allocated and freed with GDALWarpAppOptionsNew() and GDALWarpAppOptionsFree() respectively.

35.41.2 Member Data Documentation

35.41.2.1 int GDALWarpAppOptions::bCopyBandInfo

copy band information from the first source dataset

35.41.2.2 int GDALWarpAppOptions::bCopyMetadata

copy dataset and band metadata will be copied from the first source dataset. Items that differ between source datasets will be set "*" (see GDALWarpAppOptions::pszMDConflictValue)

35.41.2.3 int GDALWarpAppOptions::bCropToCutline

crop the extent of the target dataset to the extent of the cutline

35.41.2.4 bool GDALWarpAppOptions::bDisableSrcAlpha

Prevent a source alpha band from being considered as such

35.41.2.5 bool GDALWarpAppOptions::bEnableDstAlpha

creates an output alpha band to identify nodata (unset/transparent) pixels when set to true

35.41.2.6 bool GDALWarpAppOptions::bEnableSrcAlpha

forces the last band of an input file to be considered as alpha band.

35.41.2.7 int GDALWarpAppOptions::bMulti

use multithreaded warping implementation. Multiple threads will be used to process chunks of image and perform input/output operation simultaneously.

35.41.2.8 bool GDALWarpAppOptions::bNoVShiftGrid

Whether to disable vertical grid shift adjustment

35.41.2.9 int GDALWarpAppOptions::bQuiet

allow or suppress progress monitor and other non-error output

35.41.2.10 bool GDALWarpAppOptions::bSetColorInterpretation

set the color interpretation of the bands of the target dataset from the source dataset

35.41.2.11 int GDALWarpAppOptions::bTargetAlignedPixels

align the coordinates of the extent of the output file to the values of the GDALWarpAppOptions::dfXRes and GDALWarpAppOptions::dfYRes, such that the aligned extent includes the minimum extent.

35.41.2.12 double GDALWarpAppOptions::dfMinX

set georeferenced extents of output file to be created (in target SRS by default, or in the SRS specified with pszTE_SRS)

35.41.2.13 double GDALWarpAppOptions::dfWarpMemoryLimit

the amount of memory (in megabytes) that the warp API is allowed to use for caching.

35.41.2.14 double GDALWarpAppOptions::dfXRes

set output file resolution (in target georeferenced units)

35.41.2.15 GDALDataType GDALWarpAppOptions::eOutputType

the data type of the output bands

35.41.2.16 GDALResampleAlg GDALWarpAppOptions::eResampleAlg

the resampling method. Available methods are: near, bilinear, cubic, cubicspline, lanczos, average, mode, max, min, med, q1, q3

35.41.2.17 GDALDataType GDALWarpAppOptions::eWorkingType

working pixel data type. The data type of pixels in the source image and destination image buffers.

35.41.2.18 int GDALWarpAppOptions::nForcePixels

set output file size in pixels and lines. If GDALWarpAppOptions::nForcePixels or GDALWarpAppOptions::nForceLines is set to 0, the other dimension will be guessed from the computed resolution. Note that GDALWarpAppOptions::nForcePixels and GDALWarpAppOptions::nForceLines cannot be used with GDALWarpAppOptions::dfXRes and GDALWarpAppOptions::dfYRes.

35.41.2.19 int GDALWarpAppOptions::nOvLevel

overview level of source files to be used

35.41.2.20 char** GDALWarpAppOptions::papszCreateOptions

list of create options for the output format driver. See format specific documentation for legal creation options for each format.

35.41.2.21 char** GDALWarpAppOptions::papszTO

list of transformer options suitable to pass to GDALCreateGenImgProjTransformer2(). ("NAME1=VALUE1","NAME2=VALUE2",...)

35.41.2.22 char** GDALWarpAppOptions::papszWarpOptions

list of warp options. ("NAME1=VALUE1","NAME2=VALUE2",...). The GDALWarpOptions::papszWarpOptions docs show all options.

35.41.2.23 GDALProgressFunc GDALWarpAppOptions::pfnProgress

the progress function to use

35.41.2.24 void* GDALWarpAppOptions::pProgressData

pointer to the progress data variable

35.41.2.25 char* GDALWarpAppOptions::pszCLayer

the named layer to be selected from the cutline datasource

35.41.2.26 char* GDALWarpAppOptions::pszCSQL

SQL query to select the cutline features instead of from a layer with pszCLayer

35.41.2.27 char* GDALWarpAppOptions::pszCutlineDSName

enable use of a blend cutline from the name OGR support pszCutlineDSName

35.41.2.28 char* GDALWarpAppOptions::pszCWHERE

restrict desired cutline features based on attribute query

35.41.2.29 char* GDALWarpAppOptions::pszDstNodata

nodata values for output bands (different values can be supplied for each band). ("value1 value2 ..."). New files will be initialized to this value and if possible the nodata value will be recorded in the output file. Use a value of "None" to ensure that nodata is not defined. If this argument is not used then nodata values will be copied from the source dataset.

35.41.2.30 char* GDALWarpAppOptions::pszFormat

output format. The default is GeoTIFF (GTiff). Use the short format name.

35.41.2.31 char* GDALWarpAppOptions::pszMDConflictValue

value to set metadata items that conflict between source datasets (default is "*"). Use "" to remove conflicting items.

35.41.2.32 char* GDALWarpAppOptions::pszSrcNodata

nodata masking values for input bands (different values can be supplied for each band). ("value1 value2 ..."). Masked values will not be used in interpolation. Use a value of "None" to ignore intrinsic nodata settings on the source dataset.

35.41.2.33 char* GDALWarpAppOptions::pszTE_SRS

the SRS in which to interpret the coordinates given in GDALWarpAppOptions::dfMinX, GDALWarpAppOptions::dfMinY, GDALWarpAppOptions::dfMaxX and GDALWarpAppOptions::dfMaxY. The SRS may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT. It is a convenience e.g. when knowing the output coordinates in a geodetic long/lat SRS, but still wanting a result in a projected coordinate system.

The documentation for this struct was generated from the following file:

- gdalwarp_lib.cpp

35.42 GDALWarpAppOptionsForBinary Struct Reference

Public Attributes

- char ** **papszSrcFiles**
- char * **pszDstFilename**
- int **bQuiet**
- char ** **papszOpenOptions**
- char ** papszDestOpenOptions
- int **bOverwrite**
- int **bCreateOutput**
- int **bFormatExplicitlySet**
- char * **pszFormat**

35.42.1 Member Data Documentation

35.42.1.1 char** GDALWarpAppOptionsForBinary::papszDestOpenOptions

output dataset open option (format specific)

The documentation for this struct was generated from the following file:

- gdal_utils_priv.h

35.43 Gradient< T, alg > Struct Template Reference

Static Public Member Functions

- static void **calc** (const T *afWin, double inv_ewres, double inv_nsres, double &x, double &y)

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.44 Gradient< T, HORN > Struct Template Reference

Static Public Member Functions

- static void **calc** (const T *afWin, double inv_ewres, double inv_nsres, double &x, double &y)

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.45 Gradient< T, ZEVENBERGEN_THORNE > Struct Template Reference

Static Public Member Functions

- static void **calc** (const T *afWin, double inv_ewres, double inv_nsres, double &x, double &y)

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.46 LayerTranslator Class Reference

Public Member Functions

- int **Translate** (OGRFeature *poFeatureIn, TargetLayerInfo *psInfo, GIntBig nCountLayerFeatures, GIntBig *pnReadFeatureCount, GIntBig &nTotalEventsDone, GDALProgressFunc pfnProgress, void *pProgressArg, GDALVectorTranslateOptions *psOptions)

Public Attributes

- GDALDataset * **m_poSrcDS**
- GDALDataset * **m_poODS**
- bool **m_bTransform**
- bool **m_bWrapDateline**
- CPLString **m_osDateLineOffset**
- OGRSpatialReference * **m_poOutputSRS**
- bool **m_bNullifyOutputSRS**
- OGRSpatialReference * **m_poUserSourceSRS**
- OGRCoordinateTransformation * **m_poGCPCoordTrans**
- int **m_eGType**
- GeomTypeConversion **m_eGeomTypeConversion**
- int **m_nCoordDim**
- GeomOperation **m_eGeomOp**
- double **m_dfGeomOpParam**
- OGRGeometry * **m_poClipSrc**
- OGRGeometry * **m_poClipDst**

- bool **m_bExplodeCollections**
- bool **m_bNativeData**
- GIntBig **m_nLimit**

The documentation for this class was generated from the following file:

- ogr2ogr_lib.cpp

35.47 ListFieldDesc Struct Reference

Public Attributes

- int **iSrcIndex**
- OGRFieldType **eType**
- int **nMaxOccurrences**
- int **nWidth**

The documentation for this struct was generated from the following file:

- ogr2ogr_lib.cpp

35.48 NamedColor Struct Reference

Public Attributes

- const char * **name**
- float **r**
- float **g**
- float **b**

The documentation for this struct was generated from the following file:

- gdaldem_lib.cpp

35.49 OGR2OGRSpatialReferenceHolder Class Reference

Public Member Functions

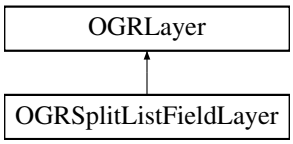
- void **assignNoRefIncrease** (OGRSpatialReference *poSRS)
- OGRSpatialReference * **get** ()

The documentation for this class was generated from the following file:

- ogr2ogr_lib.cpp

35.50 OGRSplitListFieldLayer Class Reference

Inheritance diagram for OGRSplitListFieldLayer:



Public Member Functions

- **OGRSplitListFieldLayer** (OGRLayer ∗poSrcLayer, int nMaxSplitListSubFields)
- bool **BuildLayerDefn** (GDALProgressFunc pfnProgress, void ∗pProgressArg)
- virtual OGRFeature ∗ **GetNextFeature** () override
- virtual OGRFeature ∗ **GetFeature** (GIntBig nFID) override
- virtual OGRFeatureDefn ∗ **GetLayerDefn** () override
- virtual void **ResetReading** () override
- virtual int **TestCapability** (const char ∗) override
- virtual GIntBig **GetFeatureCount** (int bForce=TRUE) override
- virtual OGRSpatialReference ∗ **GetSpatialRef** () override
- virtual OGRGeometry ∗ **GetSpatialFilter** () override
- virtual OGRStyleTable ∗ **GetStyleTable** () override
- virtual void **SetSpatialFilter** (OGRGeometry ∗poGeom) override
- virtual void **SetSpatialFilter** (int iGeom, OGRGeometry ∗poGeom) override
- virtual void **SetSpatialFilterRect** (double dfMinX, double dfMinY, double dfMaxX, double dfMaxY) override
- virtual void **SetSpatialFilterRect** (int iGeom, double dfMinX, double dfMinY, double dfMaxX, double dfMaxY) override
- virtual OGRErr **SetAttributeFilter** (const char ∗pszFilter) override

The documentation for this class was generated from the following file:

- ogr2ogr_lib.cpp

35.51 SetupTargetLayer Class Reference

Public Member Functions

- TargetLayerInfo ∗ **Setup** (OGRLayer ∗poSrcLayer, const char ∗pszNewLayerName, GDALVectorTranslate-Options ∗psOptions, GIntBig &nTotalEventsDone)

Public Attributes

- GDALDataset ∗ **m_poSrcDS**
- GDALDataset ∗ **m_poDstDS**
- char ∗∗ **m_papszLCO**
- OGRSpatialReference ∗ **m_poOutputSRS**
- bool **m_bNullifyOutputSRS**
- char ∗∗ **m_papszSelFields**
- bool **m_bAppend**
- bool **m_bAddMissingFields**
- int **m_eGType**

- GeomTypeConversion **m_eGeomTypeConversion**
- int **m_nCoordDim**
- bool **m_bOverwrite**
- char ** **m_papszFieldTypesToString**
- char ** **m_papszMapFieldType**
- bool **m_bUnsetFieldWidth**
- bool **m_bExplodeCollections**
- const char * **m_pszZField**
- char ** **m_papszFieldMap**
- const char * **m_pszWHERE**
- bool **m_bExactFieldNameMatch**
- bool **m_bQuiet**
- bool **m_bForceNullable**
- bool **m_bUnsetDefault**
- bool **m_bUnsetFid**
- bool **m_bPreserveFID**
- bool **m_bCopyMD**
- bool **m_bNativeData**
- bool **m_bNewDataSource**

The documentation for this class was generated from the following file:

- ogr2ogr_lib.cpp

35.52 TargetLayerInfo Struct Reference

Public Attributes

- OGRLayer * **poSrcLayer**
- GIntBig **nFeaturesRead**
- bool **bPerFeatureCT**
- OGRLayer * **poDstLayer**
- OGRCoordinateTransformation ** **papoCT**
- char *** **papapszTransformOptions**
- int * **panMap**
- int **iSrcZField**
- int **iSrcFIDField**
- int **iRequestedSrcGeomField**
- bool **bPreserveFID**

The documentation for this struct was generated from the following file:

- ogr2ogr_lib.cpp

35.53 ThreadContext Struct Reference

Public Attributes

- CPLJoinableThread * **hThread**
- int **bRet**

The documentation for this struct was generated from the following file:

- test_ogrsf.cpp

35.54 VRTBuilder Class Reference

Public Member Functions

- **VRTBuilder** (const char ∗pszOutputFilename, int nInputFiles, const char ∗const ∗ppszInputFilenames, GDALDatasetH ∗pahSrcDSIn, const int ∗panBandListIn, int nBandCount, int nMaxBandNo, ResolutionStrategy resolutionStrategy, double we_res, double ns_res, int bTargetAlignedPixels, double minX, double minY, double maxX, double maxY, int bSeparate, int bAllowProjectionDifference, int bAddAlpha, int bHideNoData, int nSubdataset, const char ∗pszSrcNoData, const char ∗pszVRTNoData, const char ∗pszOutputSRS, const char ∗pszResampling, const char ∗const ∗papszOpenOptionsIn)
- GDALDataset ∗ **Build** (GDALProgressFunc pfnProgress, void ∗pProgressData)

The documentation for this class was generated from the following file:

- gdalbuildvrt_lib.cpp

Chapter 36

File Documentation

36.1 gdal_utils.h File Reference

```
#include "cpl_port.h"
#include "gdal.h"
```

Typedefs

- typedef typedefCPL_C_START
  struct GDALInfoOptions GDALInfoOptions
- typedef struct
  GDALInfoOptionsForBinary GDALInfoOptionsForBinary
- typedef struct GDALTranslateOptions GDALTranslateOptions
- typedef struct
  GDALTranslateOptionsForBinary GDALTranslateOptionsForBinary
- typedef struct GDALWarpAppOptions GDALWarpAppOptions
- typedef struct
  GDALWarpAppOptionsForBinary GDALWarpAppOptionsForBinary
- typedef struct
  GDALVectorTranslateOptions GDALVectorTranslateOptions
- typedef struct
  GDALVectorTranslateOptionsForBinary GDALVectorTranslateOptionsForBinary
- typedef struct
  GDALDEMProcessingOptions GDALDEMProcessingOptions
- typedef struct
  GDALDEMProcessingOptionsForBinary GDALDEMProcessingOptionsForBinary
- typedef struct GDALNearblackOptions GDALNearblackOptions
- typedef struct
  GDALNearblackOptionsForBinary GDALNearblackOptionsForBinary
- typedef struct GDALGridOptions GDALGridOptions
- typedef struct
  GDALGridOptionsForBinary GDALGridOptionsForBinary
- typedef struct GDALRasterizeOptions GDALRasterizeOptions
- typedef struct
  GDALRasterizeOptionsForBinary GDALRasterizeOptionsForBinary
- typedef struct GDALBuildVRTOptions GDALBuildVRTOptions
- typedef struct
  GDALBuildVRTOptionsForBinary GDALBuildVRTOptionsForBinary

Functions

- GDALInfoOptions CPL_DLL * GDALInfoOptionsNew (char **papszArgv, GDALInfoOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALInfoOptionsFree (GDALInfoOptions *psOptions)
- char CPL_DLL * GDALInfo (GDALDatasetH hDataset, const GDALInfoOptions *psOptions)
- GDALTranslateOptions CPL_DLL * GDALTranslateOptionsNew (char **papszArgv, GDALTranslateOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALTranslateOptionsFree (GDALTranslateOptions *psOptions)
- void CPL_DLL GDALTranslateOptionsSetProgress (GDALTranslateOptions *psOptions, GDALProgressFunc pfnProgress, void *pProgressData)
- GDALDatasetH CPL_DLL GDALTranslate (const char *pszDestFilename, GDALDatasetH hSrcDataset, const GDALTranslateOptions *psOptions, int *pbUsageError)
- GDALWarpAppOptions CPL_DLL * GDALWarpAppOptionsNew (char **papszArgv, GDALWarpAppOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALWarpAppOptionsFree (GDALWarpAppOptions *psOptions)
- void CPL_DLL GDALWarpAppOptionsSetProgress (GDALWarpAppOptions *psOptions, GDALProgressFunc pfnProgress, void *pProgressData)
- void CPL_DLL GDALWarpAppOptionsSetWarpOption (GDALWarpAppOptions *psOptions, const char *pszKey, const char *pszValue)
- GDALDatasetH CPL_DLL GDALWarp (const char *pszDest, GDALDatasetH hDstDS, int nSrcCount, GDALDatasetH *pahSrcDS, const GDALWarpAppOptions *psOptions, int *pbUsageError)
- GDALVectorTranslateOptions CPL_DLL * GDALVectorTranslateOptionsNew (char **papszArgv, GDALVectorTranslateOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALVectorTranslateOptionsFree (GDALVectorTranslateOptions *psOptions)
- void CPL_DLL GDALVectorTranslateOptionsSetProgress (GDALVectorTranslateOptions *psOptions, GDALProgressFunc pfnProgress, void *pProgressData)
- GDALDatasetH CPL_DLL GDALVectorTranslate (const char *pszDest, GDALDatasetH hDstDS, int nSrcCount, GDALDatasetH *pahSrcDS, const GDALVectorTranslateOptions *psOptions, int *pbUsageError)
- GDALDEMProcessingOptions CPL_DLL * GDALDEMProcessingOptionsNew (char **papszArgv, GDALDEMProcessingOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALDEMProcessingOptionsFree (GDALDEMProcessingOptions *psOptions)
- void CPL_DLL GDALDEMProcessingOptionsSetProgress (GDALDEMProcessingOptions *psOptions, GDALProgressFunc pfnProgress, void *pProgressData)
- GDALDatasetH CPL_DLL GDALDEMProcessing (const char *pszDestFilename, GDALDatasetH hSrcDataset, const char *pszProcessing, const char *pszColorFilename, const GDALDEMProcessingOptions *psOptions, int *pbUsageError)
- GDALNearblackOptions CPL_DLL * GDALNearblackOptionsNew (char **papszArgv, GDALNearblackOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALNearblackOptionsFree (GDALNearblackOptions *psOptions)
- void CPL_DLL GDALNearblackOptionsSetProgress (GDALNearblackOptions *psOptions, GDALProgressFunc pfnProgress, void *pProgressData)
- GDALDatasetH CPL_DLL GDALNearblack (const char *pszDest, GDALDatasetH hDstDS, GDALDatasetH hSrcDS, const GDALNearblackOptions *psOptions, int *pbUsageError)
- GDALGridOptions CPL_DLL * GDALGridOptionsNew (char **papszArgv, GDALGridOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALGridOptionsFree (GDALGridOptions *psOptions)
- void CPL_DLL GDALGridOptionsSetProgress (GDALGridOptions *psOptions, GDALProgressFunc pfnProgress, void *pProgressData)
- GDALDatasetH CPL_DLL GDALGrid (const char *pszDest, GDALDatasetH hSrcDS, const GDALGridOptions *psOptions, int *pbUsageError)
- GDALRasterizeOptions CPL_DLL * GDALRasterizeOptionsNew (char **papszArgv, GDALRasterizeOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALRasterizeOptionsFree (GDALRasterizeOptions *psOptions)
- void CPL_DLL GDALRasterizeOptionsSetProgress (GDALRasterizeOptions *psOptions, GDALProgressFunc pfnProgress, void *pProgressData)

- GDALDatasetH CPL_DLL GDALRasterize (const char *pszDest, GDALDatasetH hDstDS, GDALDatasetH hSrcDS, const GDALRasterizeOptions *psOptions, int *pbUsageError)
- GDALBuildVRTOptions CPL_DLL * GDALBuildVRTOptionsNew (char **papszArgv, GDALBuildVRTOptionsForBinary *psOptionsForBinary)
- void CPL_DLL GDALBuildVRTOptionsFree (GDALBuildVRTOptions *psOptions)
- void CPL_DLL GDALBuildVRTOptionsSetProgress (GDALBuildVRTOptions *psOptions, GDALProgressFunc pfnProgress, void *pProgressData)
- GDALDatasetH CPL_DLL GDALBuildVRT (const char *pszDest, int nSrcCount, GDALDatasetH *pahSrcDS, const char *const *papszSrcDSNames, const GDALBuildVRTOptions *psOptions, int *pbUsageError)

36.1.1 Detailed Description

Public (C callable) GDAL Utilities entry points.

Since

GDAL 2.1

36.1.2 Typedef Documentation

36.1.2.1 typedef struct GDALBuildVRTOptions GDALBuildVRTOptions

Options for GDALBuildVRT(). Opaque type

36.1.2.2 typedef struct GDALBuildVRTOptionsForBinary GDALBuildVRTOptionsForBinary

Opaque type

36.1.2.3 typedef struct GDALDEMProcessingOptions GDALDEMProcessingOptions

Options for GDALDEMProcessing(). Opaque type

36.1.2.4 typedef struct GDALDEMProcessingOptionsForBinary GDALDEMProcessingOptionsForBinary

Opaque type

36.1.2.5 typedef struct GDALGridOptions GDALGridOptions

Options for GDALGrid(). Opaque type

36.1.2.6 typedef struct GDALGridOptionsForBinary GDALGridOptionsForBinary

Opaque type

36.1.2.7 typedef typedefCPL_C_START struct GDALInfoOptions GDALInfoOptions

Options for GDALInfo(). Opaque type

36.1.2.8 typedef struct GDALInfoOptionsForBinary GDALInfoOptionsForBinary

Opaque type

36.1.2.9 typedef struct GDALNearblackOptions GDALNearblackOptions

Options for GDALNearblack(). Opaque type

36.1.2.10 typedef struct GDALNearblackOptionsForBinary GDALNearblackOptionsForBinary

Opaque type

36.1.2.11 typedef struct GDALRasterizeOptions GDALRasterizeOptions

Options for GDALRasterize(). Opaque type

36.1.2.12 typedef struct GDALRasterizeOptionsForBinary GDALRasterizeOptionsForBinary

Opaque type

36.1.2.13 typedef struct GDALTranslateOptions GDALTranslateOptions

Options for GDALTranslate(). Opaque type

36.1.2.14 typedef struct GDALTranslateOptionsForBinary GDALTranslateOptionsForBinary

Opaque type

36.1.2.15 typedef struct GDALVectorTranslateOptions GDALVectorTranslateOptions

Options for GDALVectorTranslate(). Opaque type

36.1.2.16 typedef struct GDALVectorTranslateOptionsForBinary GDALVectorTranslateOptionsForBinary

Opaque type

36.1.2.17 typedef struct GDALWarpAppOptions GDALWarpAppOptions

Options for GDALWarp(). Opaque type

36.1.2.18 typedef struct GDALWarpAppOptionsForBinary GDALWarpAppOptionsForBinary

Opaque type

36.1.3 Function Documentation

36.1.3.1 GDALDatasetH CPL_DLL GDALBuildVRT (const char * *pszDest*, int *nSrcCount*, GDALDatasetH * *pahSrcDS*, const char *const * *papszSrcDSNames*, const GDALBuildVRTOptions * *psOptionsIn*, int * *pbUsageError*)

Build a VRT from a list of datasets.

This is the equivalent of the `gdalbuildvrt` utility.

GDALBuildVRTOptions* must be allocated and freed with GDALBuildVRTOptionsNew() and GDALBuildVRTOptionsFree() respectively. pahSrcDS and papszSrcDSNames cannot be used at the same time.

Parameters

pszDest	the destination dataset path.
nSrcCount	the number of input datasets.
pahSrcDS	the list of input datasets (or NULL, exclusive with papszSrcDSNames)
papszSrcDS-Names	the list of input dataset names (or NULL, exclusive with pahSrcDS)
psOptionsIn	the options struct returned by GDALBuildVRTOptionsNew() or NULL.
pbUsageError	the pointer to int variable to determine any usage error has occurred.

Returns

the output dataset (new dataset that must be closed using GDALClose()) or NULL in case of error.

Since

GDAL 2.1

36.1.3.2 void CPL_DLL GDALBuildVRTOptionsFree (GDALBuildVRTOptions * *psOptions*)

Frees the GDALBuildVRTOptions struct.

Parameters

psOptions	the options struct for GDALBuildVRT().

Since

GDAL 2.1

36.1.3.3 GDALBuildVRTOptions CPL_DLL* GDALBuildVRTOptionsNew (char ** *papszArgv,* GDALBuildVRTOptionsForBinary * *psOptionsForBinary*)

Allocates a GDALBuildVRTOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `gdalbuildvrt` utility.
psOptionsFor-Binary	(output) may be NULL (and should generally be NULL), otherwise (gdal_translate_bin.cpp use case) must be allocated with GDALBuildVRTOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options,...

Returns

pointer to the allocated GDALBuildVRTOptions struct. Must be freed with GDALBuildVRTOptionsFree().

Since

GDAL 2.1

36.1.3.4 void CPL_DLL GDALBuildVRTOptionsSetProgress (GDALBuildVRTOptions * *psOptions,* GDALProgressFunc *pfnProgress,* void * *pProgressData*)

Set a progress function.

Parameters

psOptions	the options struct for GDALBuildVRT().
pfnProgress	the progress callback.
pProgressData	the user data for the progress callback.

Since

GDAL 2.1

36.1.3.5 GDALDatasetH CPL_DLL GDALDEMProcessing (const char * *pszDest*, GDALDatasetH *hSrcDataset*, const char * *pszProcessing*, const char * *pszColorFilename*, const GDALDEMProcessingOptions * *psOptionsIn*, int * *pbUsageError*)

Apply a DEM processing.

This is the equivalent of the `gdaldem` utility.

GDALDEMProcessingOptions* must be allocated and freed with GDALDEMProcessingOptionsNew() and GDALDEMProcessingOptionsFree() respectively.

Parameters

pszDest	the destination dataset path.
hSrcDataset	the source dataset handle.
pszProcessing	the processing to apply (one of "hillshade", "slope", "aspect", "color-relief", "TRI", "TPI", "Roughness")
pszColorFilename	color file (mandatory for "color-relief" processing, should be NULL otherwise)
psOptionsIn	the options struct returned by GDALDEMProcessingOptionsNew() or NULL.
pbUsageError	the pointer to int variable to determine any usage error has occurred or NULL.

Returns

the output dataset (new dataset that must be closed using GDALClose()) or NULL in case of error.

Since

GDAL 2.1

36.1.3.6 void CPL_DLL GDALDEMProcessingOptionsFree (GDALDEMProcessingOptions * *psOptions*)

Frees the GDALDEMProcessingOptions struct.

Parameters

psOptions	the options struct for GDALDEMProcessing().

Since

GDAL 2.1

36.1.3.7 GDALDEMProcessingOptions CPL_DLL* GDALDEMProcessingOptionsNew (char ** *papszArgv*, GDALDEMProcessingOptionsForBinary * *psOptionsForBinary*)

Allocates a GDALDEMProcessingOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `gdaldem` utility.
psOptionsForBinary	(output) may be NULL (and should generally be NULL), otherwise (gdal_translate_bin.cpp use case) must be allocated with GDALDEMProcessingOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options,...

Returns

pointer to the allocated GDALDEMProcessingOptions struct. Must be freed with GDALDEMProcessingOptionsFree().

Since

GDAL 2.1

36.1.3.8 void CPL_DLL GDALDEMProcessingOptionsSetProgress (GDALDEMProcessingOptions * *psOptions*, GDALProgressFunc *pfnProgress*, void * *pProgressData*)

Set a progress function.

Parameters

psOptions	the options struct for GDALDEMProcessing().
pfnProgress	the progress callback.
pProgressData	the user data for the progress callback.

Since

GDAL 2.1

36.1.3.9 GDALDatasetH CPL_DLL GDALGrid (const char * *pszDest*, GDALDatasetH *hSrcDataset*, const GDALGridOptions * *psOptionsIn*, int * *pbUsageError*)

Create raster from the scattered data.

This is the equivalent of the `gdal_grid` utility.

GDALGridOptions* must be allocated and freed with GDALGridOptionsNew() and GDALGridOptionsFree() respectively.

Parameters

pszDest	the destination dataset path.
hSrcDataset	the source dataset handle.
psOptionsIn	the options struct returned by GDALGridOptionsNew() or NULL.
pbUsageError	the pointer to int variable to determine any usage error has occurred or NULL.

Returns

the output dataset (new dataset that must be closed using GDALClose()) or NULL in case of error.

Since

GDAL 2.1

36.1.3.10 void CPL_DLL GDALGridOptionsFree (GDALGridOptions * *psOptions*)

Frees the GDALGridOptions struct.

Parameters

psOptions	the options struct for GDALGrid().

Since

GDAL 2.1

36.1.3.11 GDALGridOptions CPL_DLL* GDALGridOptionsNew (char ** *papszArgv*, GDALGridOptionsForBinary * *psOptionsForBinary*)

Allocates a GDALGridOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `gdal_translate` utility.
psOptionsForBinary	(output) may be NULL (and should generally be NULL), otherwise (gdal_translate_bin.cpp use case) must be allocated with GDALGridOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options,...

Returns

pointer to the allocated GDALGridOptions struct. Must be freed with GDALGridOptionsFree().

Since

GDAL 2.1

36.1.3.12 void CPL_DLL GDALGridOptionsSetProgress (GDALGridOptions * *psOptions*, GDALProgressFunc *pfnProgress*, void * *pProgressData*)

Set a progress function.

Parameters

psOptions	the options struct for GDALGrid().
pfnProgress	the progress callback.
pProgressData	the user data for the progress callback.

Since

GDAL 2.1

36.1.3.13 char CPL_DLL* GDALInfo (GDALDatasetH *hDataset*, const GDALInfoOptions * *psOptions*)

Lists various information about a GDAL supported raster dataset.

This is the equivalent of the `gdalinfo` utility.

GDALInfoOptions* must be allocated and freed with GDALInfoOptionsNew() and GDALInfoOptionsFree() respectively.

Parameters

hDataset	the dataset handle.
psOptions	the options structure returned by GDALInfoOptionsNew() or NULL.

Returns

string corresponding to the information about the raster dataset (must be freed with CPLFree()), or NULL in case of error.

Since

GDAL 2.1

36.1.3.14 void CPL_DLL GDALInfoOptionsFree (GDALInfoOptions * *psOptions*)

Frees the GDALInfoOptions struct.

Parameters

psOptions	the options struct for GDALInfo().

Since

GDAL 2.1

36.1.3.15 GDALInfoOptions CPL_DLL* GDALInfoOptionsNew (char ** *papszArgv,* GDALInfoOptionsForBinary * *psOptionsForBinary*)

Allocates a GDALInfoOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `gdalinfo` utility.
psOptionsForBinary	(output) may be NULL (and should generally be NULL), otherwise (gdalinfo_bin.cpp use case) must be allocated with GDALInfoOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options, subdataset number...

Returns

pointer to the allocated GDALInfoOptions struct. Must be freed with GDALInfoOptionsFree().

Since

GDAL 2.1

36.1.3.16 GDALDatasetH CPL_DLL GDALNearblack (const char * *pszDest,* GDALDatasetH *hDstDS,* GDALDatasetH *hSrcDataset,* const GDALNearblackOptions * *psOptionsIn,* int * *pbUsageError*)

Convert nearly black/white borders to exact value.

This is the equivalent of the `nearblack` utility.

GDALNearblackOptions* must be allocated and freed with GDALNearblackOptionsNew() and GDALNearblackOptionsFree() respectively. pszDest and hDstDS cannot be used at the same time.

In-place update (i.e. hDstDS == hSrcDataset) is possible for formats that support it, and if the dataset is opened in update mode.

Parameters

pszDest	the destination dataset path or NULL.
hDstDS	the destination dataset or NULL. Might be equal to hSrcDataset.
hSrcDataset	the source dataset handle.
psOptionsIn	the options struct returned by GDALNearblackOptionsNew() or NULL.
pbUsageError	the pointer to int variable to determine any usage error has occurred or NULL.

Returns

the output dataset (new dataset that must be closed using GDALClose(), or hDstDS is not NULL) or NULL in case of error.

Since

GDAL 2.1

36.1.3.17 void CPL_DLL GDALNearblackOptionsFree (GDALNearblackOptions * *psOptions*)

Frees the GDALNearblackOptions struct.

Parameters

psOptions	the options struct for GDALNearblack().

Since

GDAL 2.1

36.1.3.18 GDALNearblackOptions CPL_DLL* GDALNearblackOptionsNew (char ** *papszArgv,* GDALNearblackOptionsForBinary * *psOptionsForBinary*)

Allocates a GDALNearblackOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `nearblack` utility.
psOptionsForBinary	(output) may be NULL (and should generally be NULL), otherwise (gdal_translate_bin.cpp use case) must be allocated with GDALNearblackOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options,...

Returns

pointer to the allocated GDALNearblackOptions struct. Must be freed with GDALNearblackOptionsFree().

Since

GDAL 2.1

36.1.3.19 void CPL_DLL GDALNearblackOptionsSetProgress (GDALNearblackOptions * *psOptions,* GDALProgressFunc *pfnProgress,* void * *pProgressData*)

Set a progress function.

Parameters

psOptions	the options struct for GDALNearblack().
pfnProgress	the progress callback.
pProgressData	the user data for the progress callback.

Since

GDAL 2.1

36.1.3.20 GDALDatasetH CPL_DLL GDALRasterize (const char * *pszDest*, GDALDatasetH *hDstDS*, GDALDatasetH *hSrcDataset*, const GDALRasterizeOptions * *psOptionsIn*, int * *pbUsageError*)

Burns vector geometries into a raster

This is the equivalent of the `gdal_rasterize` utility.

GDALRasterizeOptions* must be allocated and freed with GDALRasterizeOptionsNew() and GDALRasterizeOptionsFree() respectively. pszDest and hDstDS cannot be used at the same time.

Parameters

pszDest	the destination dataset path or NULL.
hDstDS	the destination dataset or NULL.
hSrcDataset	the source dataset handle.
psOptionsIn	the options struct returned by GDALRasterizeOptionsNew() or NULL.
pbUsageError	the pointer to int variable to determine any usage error has occurred or NULL.

Returns

the output dataset (new dataset that must be closed using GDALClose(), or hDstDS is not NULL) or NULL in case of error.

Since

GDAL 2.1

36.1.3.21 void CPL_DLL GDALRasterizeOptionsFree (GDALRasterizeOptions * *psOptions*)

Frees the GDALRasterizeOptions struct.

Parameters

psOptions	the options struct for GDALRasterize().

Since

GDAL 2.1

36.1.3.22 GDALRasterizeOptions CPL_DLL* GDALRasterizeOptionsNew (char ** *papszArgv*, GDALRasterizeOptionsForBinary * *psOptionsForBinary*)

Allocates a GDALRasterizeOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `gdal_rasterize` utility.
psOptionsForBinary	(output) may be NULL (and should generally be NULL), otherwise (gdal_translate_bin.cpp use case) must be allocated with GDALRasterizeOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options,...

Returns

pointer to the allocated GDALRasterizeOptions struct. Must be freed with GDALRasterizeOptionsFree().

Since

GDAL 2.1

36.1.3.23 void CPL_DLL GDALRasterizeOptionsSetProgress (GDALRasterizeOptions * *psOptions*, GDALProgressFunc *pfnProgress*, void * *pProgressData*)

Set a progress function.

Parameters

psOptions	the options struct for GDALRasterize().
pfnProgress	the progress callback.
pProgressData	the user data for the progress callback.

Since

GDAL 2.1

36.1.3.24 GDALDatasetH CPL_DLL GDALTranslate (const char * *pszDest*, GDALDatasetH *hSrcDataset*, const GDALTranslateOptions * *psOptionsIn*, int * *pbUsageError*)

Converts raster data between different formats.

This is the equivalent of the `gdal_translate` utility.

GDALTranslateOptions* must be allocated and freed with GDALTranslateOptionsNew() and GDALTranslateOptionsFree() respectively.

Parameters

pszDest	the destination dataset path.
hSrcDataset	the source dataset handle.
psOptionsIn	the options struct returned by GDALTranslateOptionsNew() or NULL.
pbUsageError	the pointer to int variable to determine any usage error has occurred or NULL.

Returns

the output dataset (new dataset that must be closed using GDALClose()) or NULL in case of error.

Since

GDAL 2.1

36.1.3.25 void CPL_DLL GDALTranslateOptionsFree (GDALTranslateOptions * *psOptions*)

Frees the GDALTranslateOptions struct.

Parameters

psOptions	the options struct for GDALTranslate().

Since

GDAL 2.1

36.1.3.26 GDALTranslateOptions CPL_DLL* GDALTranslateOptionsNew (char ** papszArgv, GDALTranslateOptionsForBinary * psOptionsForBinary)

Allocates a GDALTranslateOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `gdal_translate` utility.
psOptionsForBinary	(output) may be NULL (and should generally be NULL), otherwise (gdal_translate_bin.cpp use case) must be allocated with GDALTranslateOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options,...

Returns

pointer to the allocated GDALTranslateOptions struct. Must be freed with GDALTranslateOptionsFree().

Since

GDAL 2.1

36.1.3.27 void CPL_DLL GDALTranslateOptionsSetProgress (GDALTranslateOptions * psOptions, GDALProgressFunc pfnProgress, void * pProgressData)

Set a progress function.

Parameters

psOptions	the options struct for GDALTranslate().
pfnProgress	the progress callback.
pProgressData	the user data for the progress callback.

Since

GDAL 2.1

36.1.3.28 GDALDatasetH CPL_DLL GDALVectorTranslate (const char * pszDest, GDALDatasetH hDstDS, int nSrcCount, GDALDatasetH * pahSrcDS, const GDALVectorTranslateOptions * psOptionsIn, int * pbUsageError)

Converts vector data between file formats.

This is the equivalent of the `ogr2ogr` utility.

GDALVectorTranslateOptions* must be allocated and freed with GDALVectorTranslateOptionsNew() and GDALVectorTranslateOptionsFree() respectively. pszDest and hDstDS cannot be used at the same time.

Parameters

pszDest	the destination dataset path or NULL.
hDstDS	the destination dataset or NULL.
nSrcCount	the number of input datasets (only 1 supported currently)
pahSrcDS	the list of input datasets.
psOptionsIn	the options struct returned by GDALVectorTranslateOptionsNew() or NULL.
pbUsageError	the pointer to int variable to determine any usage error has occurred

Returns

the output dataset (new dataset that must be closed using GDALClose(), or hDstDS is not NULL) or NULL in case of error.

Since

GDAL 2.1

36.1.3.29 void CPL_DLL GDALVectorTranslateOptionsFree (GDALVectorTranslateOptions * *psOptions*)

Frees the GDALVectorTranslateOptions struct.

Parameters

psOptions	the options struct for GDALVectorTranslate().

Since

GDAL 2.1

36.1.3.30 GDALVectorTranslateOptions CPL_DLL* GDALVectorTranslateOptionsNew (char ** *papszArgv,* GDALVectorTranslateOptionsForBinary * *psOptionsForBinary*)

allocates a GDALVectorTranslateOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `ogr2ogr` utility.
psOptionsForBinary	(output) may be NULL (and should generally be NULL), otherwise (gdal_translate_bin.cpp use case) must be allocated with GDALVectorTranslateOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options,...

Returns

pointer to the allocated GDALVectorTranslateOptions struct. Must be freed with GDALVectorTranslateOptionsFree().

Since

GDAL 2.1

36.1.3.31 void CPL_DLL GDALVectorTranslateOptionsSetProgress (GDALVectorTranslateOptions * *psOptions,* GDALProgressFunc *pfnProgress,* void * *pProgressData*)

Set a progress function.

Parameters

psOptions	the options struct for GDALVectorTranslate().
pfnProgress	the progress callback.
pProgressData	the user data for the progress callback.

Since

GDAL 2.1

36.1.3.32 GDALDatasetH CPL_DLL GDALWarp (const char * *pszDest*, GDALDatasetH *hDstDS*, int *nSrcCount*, GDALDatasetH * *pahSrcDS*, const **GDALWarpAppOptions** * *psOptionsIn*, int * *pbUsageError*)

Image reprojection and warping function.

This is the equivalent of the `gdalwarp` utility.

GDALWarpAppOptions* must be allocated and freed with GDALWarpAppOptionsNew() and GDALWarpAppOptionsFree() respectively. pszDest and hDstDS cannot be used at the same time.

Parameters

pszDest	the destination dataset path or NULL.
hDstDS	the destination dataset or NULL.
nSrcCount	the number of input datasets.
pahSrcDS	the list of input datasets.
psOptionsIn	the options struct returned by GDALWarpAppOptionsNew() or NULL.
pbUsageError	the pointer to int variable to determine any usage error has occurred.

Returns

the output dataset (new dataset that must be closed using GDALClose(), or hDstDS if not NULL) or NULL in case of error.

Since

GDAL 2.1

36.1.3.33 void CPL_DLL GDALWarpAppOptionsFree (**GDALWarpAppOptions** * *psOptions*)

Frees the GDALWarpAppOptions struct.

Parameters

psOptions	the options struct for GDALWarp().

Since

GDAL 2.1

36.1.3.34 GDALWarpAppOptions CPL_DLL* GDALWarpAppOptionsNew (char ** *papszArgv*, **GDALWarpAppOptionsForBinary** * *psOptionsForBinary*)

Allocates a GDALWarpAppOptions struct.

Parameters

papszArgv	NULL terminated list of options (potentially including filename and open options too), or NULL. The accepted options are the ones of the `gdalwarp` utility.
psOptionsForBinary	(output) may be NULL (and should generally be NULL), otherwise (gdal_translate_bin.cpp use case) must be allocated with GDALWarpAppOptionsForBinaryNew() prior to this function. Will be filled with potentially present filename, open options,...

Returns

pointer to the allocated GDALWarpAppOptions struct. Must be freed with GDALWarpAppOptionsFree().

Since

GDAL 2.1

36.1.3.35 void CPL_DLL GDALWarpAppOptionsSetProgress (GDALWarpAppOptions * *psOptions*, GDALProgressFunc *pfnProgress*, void * *pProgressData*)

Set a progress function.

Parameters

psOptions	the options struct for GDALWarpApp().
pfnProgress	the progress callback.
pProgressData	the user data for the progress callback.

Since

GDAL 2.1

36.1.3.36 void CPL_DLL GDALWarpAppOptionsSetWarpOption (GDALWarpAppOptions * *psOptions*, const char * *pszKey*, const char * *pszValue*)

Set a warp option

Parameters

psOptions	the options struct for GDALWarpApp().
pszKey	key.
pszValue	value.

Since

GDAL 2.1