Generated on Tue Aug 19 12:18:23 2014 by Doxygen] Generated on Tue Aug 19 12:18:23 2014 by Doxygen

Reference Manual

Contents

Chapter 1

GDAL Utilities

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)
- gdal-config - Get options required to build software using GDAL.

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 commandline. 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>" commandline option but the web page for the format is the definitive source of information on driver creation options.

-a_srs *SRS* Several utilities, (gdal_translate and gdalwarp) include the ability to specify coordinate systems with commandline 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 commandline 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-paci
- *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 (ie. supports CreateCopy); 'rw+' is read, write and update (ie. supports Create). A 'v' is appended for formats supporting virtual IO (/vsimem, /vsigzip, /vsizip, etc). 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 commandline 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 commandline options and exit.

Chapter 2

gdalinfo

lists information about a raster dataset

2.1 SYNOPSIS

```
gdalinfo [--help-general] [-mm] [-stats] [-hist] [-nogcp] [-nomd]
         [-noct] [-nofl] [-checksum] [-proj4] [-mdd domain]*
 [-sd subdataset] datasetname
```

2.2 DESCRIPTION

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

-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.

-noct Suppress printing of color table.

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

-mdd domain Report metadata for the specified domain

-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.

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 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[%] ysize[%]]
       [-unscale] [-scale [src_min src_max [dst_min dst_max]]]
       [-srcwin xoff yoff xsize ysize] [-projwin ulx uly lrx lry]
       [-a_srs srs_def] [-a_ullr ulx uly lrx lry] [-a_nodata value]
       [-gcp pixel line easting northing [elevation]]*
       [-mo "META-TAG=VALUE"]* [-q] [-sds]
       [-co "NAME=VALUE"]* [-stats]
       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: Do'nt 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 1st 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 explicitely used to ben 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. Usefull 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[%] ysize[%]*****:** 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.

-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.

-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.

-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.

-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

-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.

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

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 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 --conf
```

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,average_mp,average_magphase,mode}]
         [-ro] [-clean] [--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 over several downsampling algorithms.

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

-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.

***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 4x4 approximate 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.

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 independantly 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.

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. (eg --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. (eg

--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 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 [-tileindex field_name] [-write_absolute_path] [-skip_different_projection] index_file [gdal_fi
```

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.

- The shapefile (index_file) will be created if it doesn't already exist, otherwise it will append to the existing file.
- The default tile index field is 'location'.
- Raster filenames will be put in the file exactly as they are specified on the commandline unless the option -write_absolute_path is used.
- If -skip_different_projection is specified, only files with same projection ref as files already inserted in the tileindex will be inserted.
- Simple rectangular polygons are generated in the same coordinate system as the rasters.

5.3 EXAMPLE

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

Chapter 6

gdalbuildvrt

Builds a VRT from a list of datasets. (compiled by default since GDAL 1.6.1)

6.1 SYNOPSIS

```
gdalbuildvrt [-tileindex field_name] [-resolution {highest|lowest|average|user}]
             [-tr xres yres] [-tap] [-separate] [-allow_projection_difference] [-q]
             [-te xmin ymin xmax ymax] [-addalpha] [-hidenodata]
             [-srcnodata "value [value...]"] [-vrtnodata "value [value...]"]
             [-input_file_list my_liste.txt] [-overwrite] output.vrt [gdalfile]*
```

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 *stacked* 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 may depend on the order they are inserted in the VRT file, but this behaviour should not be relied on.

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 instrinsic 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.

-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, instrinsic 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 *stacked* 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.

-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

```
gdalbuildvrt doq_index.vrt doq/*.tif
gdalbuildvrt -input_file_list my_liste.txt doq_index.vrt
gdalbuildvrt -separate rgb.vrt red.tif green.tif blue.tif
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] [-f <formatname>] [-i <interval>]
                    [-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.

-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 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]
       [-l layername]* [-where expression] [-sql select_statement]
       [-of format] [-a_srs srs_def] [-co "NAME=VALUE"]*
       [-a_nodata value] [-init value]*
       [-te xmin ymin xmax ymax] [-tr xres yres] [-tap] [-ts width height]
       [-ot {Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/
             CInt16/CInt32/CFloat32/CFloat64}] [-q]
       <src_datasource> <dst_filename>
```

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 the vector data must in the same coordinate system as the raster data; on the fly reprojection is not provided.

Since GDAL 1.8.0, the target GDAL file can be created by gdal_rasterize. One of -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 a polygon.

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

-burn *value*: 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. These values are adjusted by the burn value given by "-burn value" or "-a attribute_name" if provided. As of now, only points and lines are drawn in 3D.

-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.

-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 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 consistant 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 pseudocolor 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 simpliest 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 pseudocolor 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]]*
    [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 (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf 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 (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf 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)

***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 Reprojection 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 airphotos 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 examle).

-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.

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] [-v] [-pct]
              [-ul_lr ulx uly lrx lry] [-n nodata_value] [-init "value [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.

-ot *datatype*: Force the output image bands to have a specific type. Use type names (ie. 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 *stacked* band.

-pct: Grab a pseudocolor table from the first input image, and use it for the output. Merging pseudocolored 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 initalized 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 [-title "Title"] [-publishurl http://yourserver/dir/]
              [-nogooglemaps] [-noopenlayers] [-nokml]
              [-googlemapskey KEY] [-forcekml] [-v]
              input_file [output_dir]
```

14.2 DESCRIPTION

This utility generates a directory with small tiles and metadata, following OSGeo Tile Map Service Specification. Simple web pages with viewers based on Google Maps and OpenLayers 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 webbrowser. You only need to upload generated directory into a web server.

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

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

-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,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.

-y *YAHOOKEY*, --yahookey=*YAHOOKEY*: Yahoo Application ID from http://developer.yahoo.com/wsregapp/.

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

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]
               [-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 subdirs 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.

-ot *datatype*: Force the output image bands to have a specific type. Use type names (ie. 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

-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 (ie.EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf 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 propageted 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 delimter 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 subdirectory called 0, the pyramids in

subdirectories numbered 1,2,.... Within each of these directories another level of subdirectories is created, numbered from 0...n, depending of how many tile rows are needed for each level. Finally, a directory contains only the 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]
  [-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.

-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).

-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 radiuses 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 radiuses are non-zero). Default is 0.

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

17.3.2 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.

17.3.3 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).

17.4 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 radiuses 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 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 GTif
```

The next command does the same thing as the previos 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.

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

Chapter 18

gdaldem

Tools to analyze and visualize DEMs. (since GDAL 1.7.0)

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]
                [-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 :
  Scale is the ratio of vertical units to horizontal
   for Feet:Latlong use scale=370400, for Meters:LatLong use scale=111120)
```

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 litterature 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.

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 optionnaly 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 tablulations and commma 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 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 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.

18.4 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


$ 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",0.01745329251994295]]
```

Chapter 20

gdallocationinfo

raster query tool

20.1 SYNOPSIS

```
Usage: gdallocationinfo [--help-general] [-xml] [-lifonly] [-valonly]
                        [-b band]* [-l_srs srs_def] [-geoloc] [-wgs84]
                        srcfile [x y]
```

20.2 DESCRIPTION

The gdallocationinfo utility provide a mechanism to query information about a pixel given it's 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 (ie. 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.

-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.

***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 commandline, or read from stdin. More than one coordinate pair can be supplied when reading coordinatesis 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.

20.3 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 21

gdalwarp

image reprojection and warping utility

21.1 SYNOPSIS

```
gdalwarp [--help-general] [--formats]
    [-s_srs srs_def] [-t_srs srs_def] [-to "NAME=VALUE"]
    [-order n | -tps | -rpc | -geoloc] [-et err_threshold]
    [-refine_gcps tolerance [minimum_gcps]]
    [-te xmin ymin xmax ymax] [-tr xres yres] [-tap] [-ts width height]
    [-wo "NAME=VALUE"] [-ot Byte/Int16/...] [-wt Byte/Int16]
    [-srcnodata "value [value...]"] [-dstnodata "value [value...]"] -dstalpha
    [-r resampling_method] [-wm memory_in_mb] [-multi] [-q]
    [-cutline datasource] [-cl layer] [-cwhere expression]
    [-csql statement] [-cblend dist_in_pixels] [-crop_to_cutline]
    [-of format] [-co "NAME=VALUE"]* [-overwrite]
    srcfile* dstfile
```

21.2 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 (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf 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 (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf 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.

-et *err_threshold*: error threshold for transformation approximation (in pixel units - defaults to 0.125).

-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).

-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

-wo *"NAME=VALUE"*: Set a warp options. 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.

-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.

-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.

***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 accomodate 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 georeferenced units of the destination file. When outputing 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.

21.3 EXAMPLE

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

Chapter 22

OGR Utility 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

Chapter 23

ogrinfo

lists information about an OGR supported data source

23.1 SYNOPSIS

```
ogrinfo [--help-general] [-ro] [-q] [-where restricted_where]
        [-spat xmin ymin xmax ymax] [-fid fid]
        [-sql statement] [-dialect dialect] [-al] [-so] [-fields={YES/NO}]
        [-geom={YES/NO/SUMMARY}][--formats]
        datasource_name [layer [layer ...]]
```

23.2 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).

-so: Summary Only: supress 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.

-sql *statement*: Execute the indicated SQL statement and return the result.

-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.

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

-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}: (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, the geometry will be reported in full OGC WKT format. Default value is YES.

--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 layerwide 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.

23.3 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 24

ogr2ogr

converts simple features data between file formats

24.1 SYNOPSIS

```
Usage: ogr2ogr [--help-general] [-skipfailures] [-append] [-update]
               [-select field_list] [-where restricted_where]
               [-progress] [-sql <sql statement>] [-dialect dialect]
               [-preserve_fid] [-fid FID]
               [-spat xmin ymin xmax ymax]
               [-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] [layer [layer ...]]

Advanced options :
               [-gt n]
               [-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]
               [[-simplify tolerance] | [-segmentize max_dist]]
               [-fieldTypeToString All|(type1[,type2]*)]
               [-splitlistfields] [-maxsubfields val]
               [-explodecollections] [-zfield field_name]
```

24.2 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.)

-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.

-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.

-where*restricted_where*: Attribute query (like SQL WHERE)

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

-spat*xmin ymin xmax ymax*: spatial query extents. Only features whose geometry intersects the extents will be selected. The geometries will not be clipped unless -clipsrc is specified

-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. Add "25D" to the name to get 2.5D versions.

-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.

-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.

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

Advanced options :

-gt *n*: group *n* features per transaction (default 200). Increase the value for better performance when writing into DBMS drivers that have transaction support.

-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)

-simplify*tolerance*: (starting with GDAL 1.9.0) distance tolerance for simplification. This method will preserve topology, in particular for polygon geometries.

-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, Real, String, Date, Time, DateTime, Binary, IntegerList, 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.

-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

24.3 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 1024** usually ensures a noticeable performance boost; defining an even bigger **-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 significantly insertion performance boot. See the PG driver documentation page.

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

24.4 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
```

More examples are given in the individual format pages.

Chapter 25

ogrtindex

creates a tileindex

25.1 SYNOPSIS

```
ogrtindex [-lnum n]... [-lname name]... [-f output_format]
          [-write_absolute_path] [-skip_different_projection]
                output_dataset src_dataset...
```

25.2 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.

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.

It is a flaw of the current ogrtindex program that no attempt is made to copy the coordinate system definition from the source datasets to the tile index (as is expected by MapServer when PROJECTION AUTO is in use).

25.3 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 26

Class Index

26.1 Class List

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

Chapter 27

Class Documentation

27.1 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.cpp

27.2 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.cpp

27.3 CutlineTransformer Class Reference

Public Member Functions

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

Public Attributes

- void ∗ **hSrcImageTransformer**

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

- gdalwarp.cpp

27.4 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.cpp

27.5 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

27.6 GDALAspectAlgData Struct Reference

Public Attributes

- int **bAngleAsAzimuth**

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

- gdaldem.cpp

27.7 GDALColorReliefDataset Class Reference

Public Member Functions

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

Friends

- class GDALColorReliefRasterBand

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

- gdaldem.cpp

27.8 GDALColorReliefRasterBand Class Reference

Public Member Functions

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

Friends

- class GDALColorReliefDataset

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

- gdaldem.cpp

27.9 GDALGeneric3x3Dataset Class Reference

Public Member Functions

- **GDALGeneric3x3Dataset** (GDALDatasetH hSrcDS, GDALRasterBandH hSrcBand, GDALDataType eDstDataType, int bDstHasNoData, double dfDstNoDataValue, GDALGeneric3x3ProcessingAlg pfnAlg, void ∗pAlgData, int bComputeAtEdges)
- CPLErr **GetGeoTransform** (double ∗padfGeoTransform)
- const char ∗ **GetProjectionRef** ()

Friends

- class GDALGeneric3x3RasterBand

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

- gdaldem.cpp

27.10 GDALGeneric3x3RasterBand Class Reference

Public Member Functions

- **GDALGeneric3x3RasterBand** (GDALGeneric3x3Dataset *poDS, GDALDataType eDstDataType)
- virtual CPLErr **IReadBlock** (int, int, void *)
- virtual double **GetNoDataValue** (int *pbHasNoData)

Friends

- class GDALGeneric3x3Dataset

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

- gdaldem.cpp

27.11 GDALHillshadeAlgData Struct Reference

Public Attributes

- double **nsres**
- double **ewres**
- double **sin_altRadians**
- double **cos_altRadians_mul_z_scale_factor**
- double **azRadians**
- double **square_z_scale_factor**

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

- gdaldem.cpp

27.12 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.cpp

27.13 ListFieldDesc Struct Reference

Public Attributes

- int **iSrcIndex**
- OGRFieldType **eType**
- int **nMaxOccurences**
- int **nWidth**

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

- ogr2ogr.cpp

27.14 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.cpp

27.15 OGRSplitListFieldLayer Class Reference

Public Member Functions

- **OGRSplitListFieldLayer** (OGRLayer ∗poSrcLayer, int nMaxSplitListSubFields)
- int **BuildLayerDefn** (GDALProgressFunc pfnProgress, void ∗pProgressArg)
- virtual OGRFeature ∗ **GetNextFeature** ()
- virtual OGRFeature ∗ **GetFeature** (long nFID)
- virtual OGRFeatureDefn ∗ **GetLayerDefn** ()
- virtual void **ResetReading** ()
- virtual int **TestCapability** (const char ∗)
- virtual int **GetFeatureCount** (int bForce=TRUE)
- virtual OGRSpatialReference ∗ **GetSpatialRef** ()
- virtual OGRGeometry ∗ **GetSpatialFilter** ()
- virtual OGRStyleTable ∗ **GetStyleTable** ()
- virtual void **SetSpatialFilter** (OGRGeometry ∗poGeom)
- virtual void **SetSpatialFilterRect** (double dfMinX, double dfMinY, double dfMaxX, double dfMaxY)
- virtual OGRErr **SetAttributeFilter** (const char ∗pszFilter)

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

- ogr2ogr.cpp

27.16 VRTBuilder Class Reference

Public Member Functions

- **VRTBuilder** (const char ∗pszOutputFilename, int nInputFiles, const char ∗const ∗ppszInputFilenames, 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, const char ∗pszSrcNoData, const char ∗pszVRTNoData)
- int **Build** (GDALProgressFunc pfnProgress, void ∗pProgressData)

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

- gdalbuildvrt.cpp